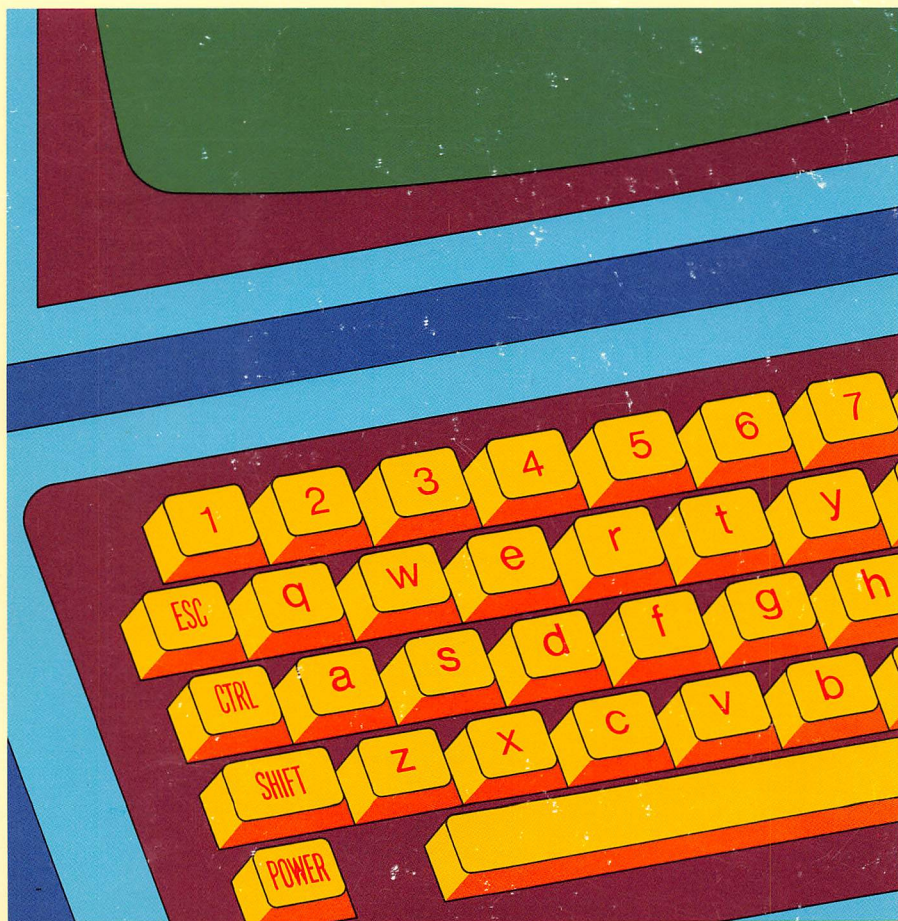




# Enhancer II

## Installation and Operation Manual





# Preliminary

First Edition  
First Printing  
6-November-1981

## Copyright Notice:

This manual and all software is Copyrighted. All rights are reserved. This document and software contained herein may not, in whole or part, be copied, photocopied, reproduced, translated or reduced to any electronic medium or machine readable form without prior consent, in writing, from Videx, Inc.

© 1981 by: Videx, Inc.  
897 NW Grant Avenue  
Corvallis, Oregon 97330  
(503) 758-0521

Enhancer ][, Videoterm, Keyboard & Display Enhancer and  
Soft Video Switch are all trademarks of Videx, Inc.

P R E L I M I N A R Y      D O C U M E N T A T I O N

Enhancer ][

T A B L E   O F   C O N T E N T S

Circa: 6-November-81

Part I

Chapter One: Introduction

- 1.a) Manual Organization and the Conventions Used
- 1.b) The Product Registration Form
- 1.c) Features and Options
- 1.d) Hardware Requirements

Chapter Two: Getting Started

- 2.a) Enhancer ][ Utilities Disc Checkout
- 2.b) How to Remove the Case From Your Apple
- 2.c) Lower Case Chip Installation and Checkout
- 2.d) Enhancer ][ Installation
- 2.e) Enhancer ][ Installation Checkout

Chapter Three: H E L P ! or What To Do If All Else Fails

- 3.a) Save Time & Money
- 3.b) Trouble-Shooting
- 3.c) The RMA Form
- 3.d) How to Package your Enhancer ][ Before Shipping It  
Back for Repair

Part II

Chapter Four: A Primer for Beginners

- 4.a) Introduction
- 4.b) Limitations of the Apple ][ Keyboard
- 4.c) What the Enhancer ][ Can Do

Chapter Five: Operation

- 5.a) Overview
- 5.b) Semantics
- 5.c) The Reset Key
- 5.d) The Caps Lock Mode
- 5.e) The Lower Case (Caps Unlock) Mode
- 5.f) User Definable Macro Keys
- 5.g) The Type Ahead Buffer
- 5.h) Self Test Diagnostics
- 5.i) Dvorak Option



## Part III

### Chapter Six: Apple ][ Language Considerations

- 6.a) Apple DOS & BASICs
- 6.b) 6502 Machine Language
- 6.c) Pascal
- 6.d) FORTRAN

### Chapter Seven: Other Software Considerations

- 7.a) Word Processors
- 7.b) CP/M
- 7.c) Other Commercial Software

### Chapter Eight: Peripheral Considerations

- 8.a) Peripherals in General
- 8.b) Videoterm
- 8.c) Language Card and Other RAM Cards
- 8.d) Softcard (Z80)
- 8.e) Printers, Parallel & Serial I/O Boards
- 8.f) Disc Drive Controllers

## Part IV

### Appendix A: The Lower Case Fix

- A.a) CAPTST
- A.b) Implementation
- A.c) The ROM Card Solution
- A.d) The RAM Card Solution
- A.e) How to Modify the Monitor Without a RAM Card
- A.f) Modifying a 2716 EPROM

### Appendix B: Lower Case Display

- B.a) Revision 0 through 6 Apples
- B.b) How to Modify Your Old Enhancer for Display Only

### Appendix C: Down Load Technical Data

### Appendix V: Tech's Installation Checklist

### Appendix W: Specifications

### Appendix X: Supporting Software

- X.a) The Configuration & Hello Programs
- X.b) The Key Filter Program
- X.c) Apple Writer Modify
- X.d) The Macro Editor
- X.e) The Down Load Program
- X.f) The OUTPATCH (Pascal) Program

### Appendix Y: Firmware Listing

### Appendix Z: Schematic

### Glossary

Notice:

Videx, Inc. reserves the right to make improvements or changes in the product described in this manual at any time without notice.

Disclaimer of All Warranties and Liability

Videx, Inc. makes no warranties, neither express nor implied except as explicitly set forth in the Limited Warranty below, with respect to this manual nor with respect to the product described in this manual, its quality, performance, merchantability or fitness for any purpose. Videx, Inc. software is sold or licensed "as is". The entire risk as to its quality and performance is with the buyer. Should the programs prove defective following their purchase, the buyer (and not Videx, Inc., its distributors, or its retailers) assumes the entire cost of all necessary servicing, repair, or correction and any incidental or consequential damages. In no event will Videx, Inc. be liable for direct, indirect, incidental, or consequential damages resulting from any defect in the hardware/software, even if Videx, Inc. has been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you.

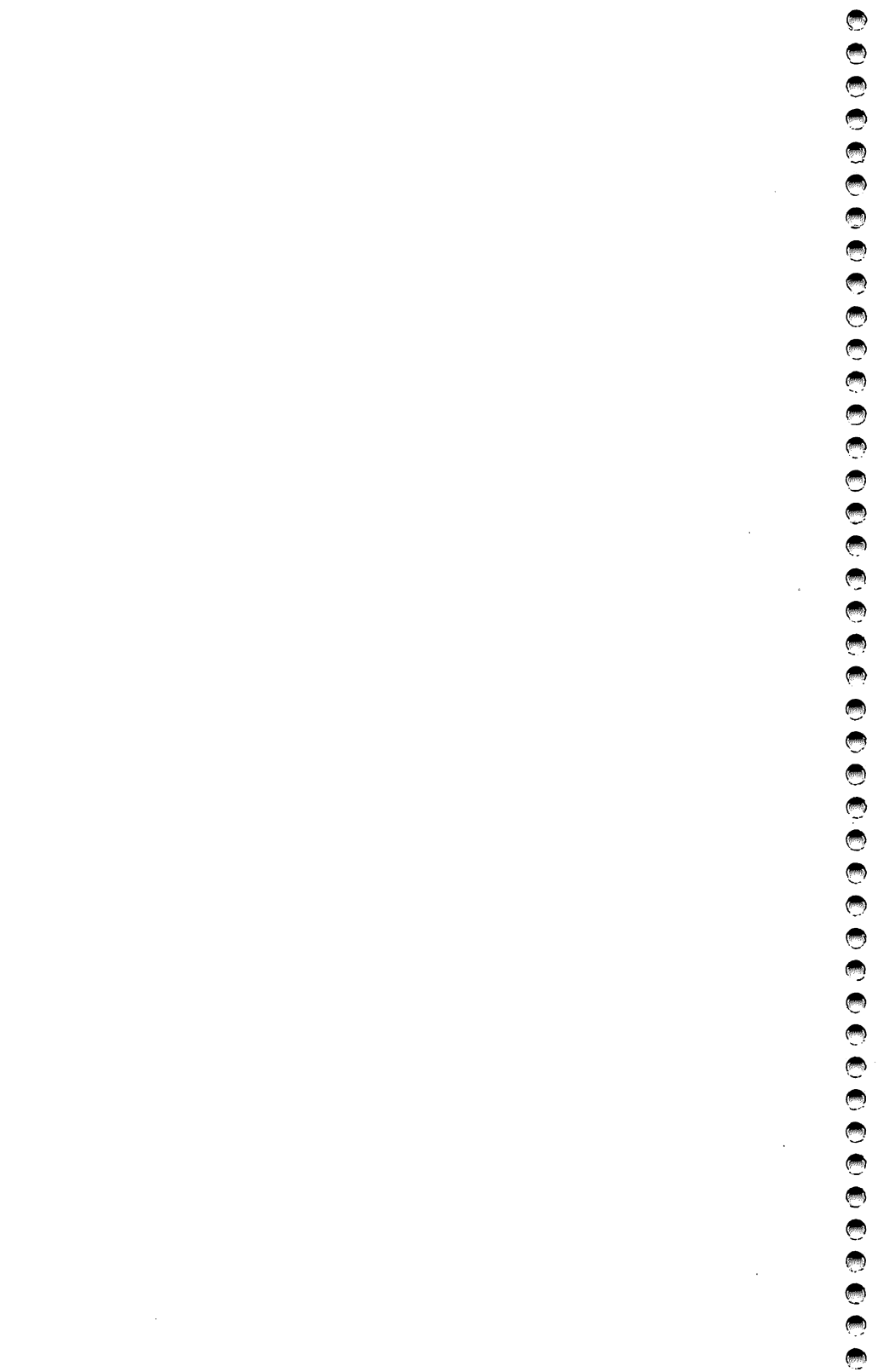
Limited Warranty:

Videx, Inc. warrants this product to be free from defects in material and workmanship for a period of ninety (90) days from the date of original purchase. Videx, Inc. agrees to repair or, at our option, replace any defective unit without charge. Videx, Inc. assumes no responsibility for any special or consequential damages. No other warranty, neither express nor implied, is authorized by Videx, Inc. Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you.

## Chapter One: Introduction

### Chapter Table of Contents.

- 1.a) Manual Organization and the Conventions Used
  - 1.a.1) Protocol
  - 1.a.2) References
  - 1.a.3) Notation
- 1.b) Product Registration Form
- 1.c) Features and Options
- 1.d) Hardware Requirements



## Chapter One: Introduction

### Section 1.a Manual Organization and the Conventions Used

The purpose of this manual is to provide technical support to the end user of the Enhancer ][ after the sale. While we have attempted to make this manual as complete as possible, it is not a replacement for the manuals to your Apple computer, peripherals or software. This edition of the manual is a preliminary edition. If you complete the product registration form, you will be notified by mail when the Second Edition is available.

The general organization is in four parts:

#### Part I

##### Preliminaries

##### Chapter One: General information

- Product registration

- Standard features

- Optional features

##### Chapter Two: Installation and initial checkout

##### Chapter Three: Troubleshooting

- How to save time and money

- Some common problems

  - (with solutions)

- The RMA form

- Warranty and non-warranty service

#### Part II

##### Operation

##### Chapters Four through Five

#### Part III

##### Interfacing

##### Chapters Six through Ten

#### Part IV

##### Quick Reference

##### Appendices: Concise technical data

This manual has been designed for ease of use by both the beginner and advanced user. The various chapters deal with certain topics in some detail, whereas the appendices contain a more technical synopsis of operation.

## Chapter One: Introduction

Chapter one is devoted to general information. Chapter two covers installation and initial checkout. Chapter three is the crisis chapter. Reading chapters two and three carefully can save both money and "system downtime". Chapter three contains a troubleshooting section and the return procedure for warranty and non-warranty repairs. Chapter four is a primer for beginners while subsequent chapters deal with the product in more detail. The appendices contain quick, concise data with references to sections containing more detailed information.

### Section 1.a.1 Protocol

The sections are numbered with a chapter.section protocol, the format of which is: chapter.section.subsection.sub-subsection.... This is family structured, i.e. each section of a chapter may have subsections, each of which may, in turn, have subsections of their own, ad infinitum. Each subsection is an expansion of one theme expressed in its parent section or subsection. Subsections may therefore be viewed as a closer or more detailed look at one topic.

Each parent section begins with a lower case letter. For example, the fifth section of chapter two would be labeled: 5.e. The first subsection of the fifth section would be: 5.e.1. The second subsection would be: 5.e.2. The first subsection of subsection of 5.e.2 would be labeled: 5.e.2.1.

The pages are numbered with a chapter - page number protocol. For example, the fifth page of chapter two would be: 2-5.

At the beginning of each chapter is a Chapter Table of Contents. Each CTOC (\see-tock\ ) completely lists all sections and subsections of that chapter. In this way, the reader can know what topics are discussed in that chapter.

### Section 1.a.2 References

References to particular pages or sections may be found within square brackets. A reference to page 2-5, for example, might appear like this:

[page: 2-5]

whereas a reference to section 2.e could appear as:

[section: 2.e].



## Chapter One: Introduction

### Section 1.a.3 Notation

Control characters will be designated with a circumflex preceding the character. For example, a control X would appear as:

`^X`

Sometimes we will use the name of the character rather than the character itself. When this is done, the name of the character - or it's abbreviation - will be contained in angle brackets. For example, a control left square bracket - `^[` - might appear as:

`<escape>`

or as:

`<esc>`

Likewise, a `^M` could be denoted as:

`<cr>`

Naturally, these characters are not meant to be entered into the computer in this long hand fashion. To enter a `^G`, for example, one would hold the CTRL (control) key down and depress the G key.

### Section 1.b The Product Registration Form

You will find a product registration form attached in the back of this manual. Another one may be found in this section. While you are not required to mail this card to us for warranty protection, it is strongly recommended that you do complete the form and mail it to us at your earliest possible convenience. This allows us to mail any information which may be of interest to our customers at a later date. It also allows us to compile data so that we may better determine the needs of all our customers and to serve you better.

Please try to answer all the questions as completely as possible. If you are unable to answer any question, leave it blank. If you have any additional comments, please use the comment card for that purpose or write us a letter. Please do not write any comments on the Product Registration Form itself since those cards are not reviewed by our technical staff. Please do NOT fold the Product Registration form.

## Chapter One: Introduction

### Section 1.c Features and Options

The following are some of the Enhancer ]['s features:

- \* Full ASCII keyboard (128 ASCII codes)
- \* The complete printable ASCII character set - including lower case - may be displayed
- \* User definable keys - with down loading from disc [section: 5.f]
- \* Type ahead buffer [section: 5.g]
- \* Auto repeat [section: 5.a.3]
- \* Fast repeat [section: 5.a.3]
- \* Normal Apple ][ mode [section: 5.d]
- \* Typewriter like operation [section: 5.e]
- \* Shift-lock feature [section: 5.e.1]
- \* Control - Reset protection [section: 5.c.1]
- \* Simple installation [chapter two]
- \* Microprocessor controlled (6500 series)
- \* Self test diagnostics [section: 5.h]
- \* A 2716 EPROM is used for on board firmware
- \* Complete firmware listings [appendix Y]
- \* Complete schematic [appendix Z]
- \* Dvorak keyboard option [section: 5.i]

Your new keyboard Enhancer ][ utilizes a sophisticated microprocessor with its own RAM. Although the Enhancer ][ is more sophisticated than it's predecessor, the Keyboard & Display Enhancer, it is significantly easier to install. The microprocessor extends the features of the Enhancer ][ beyond those of the original Keyboard & Display Enhancer with features such as user definable keys and a type ahead buffer. The caps lock, caps unlock, shift lock and control - reset features are preserved, thus supporting the normal Apple ][ keyboard as well as a typewriter like mode.

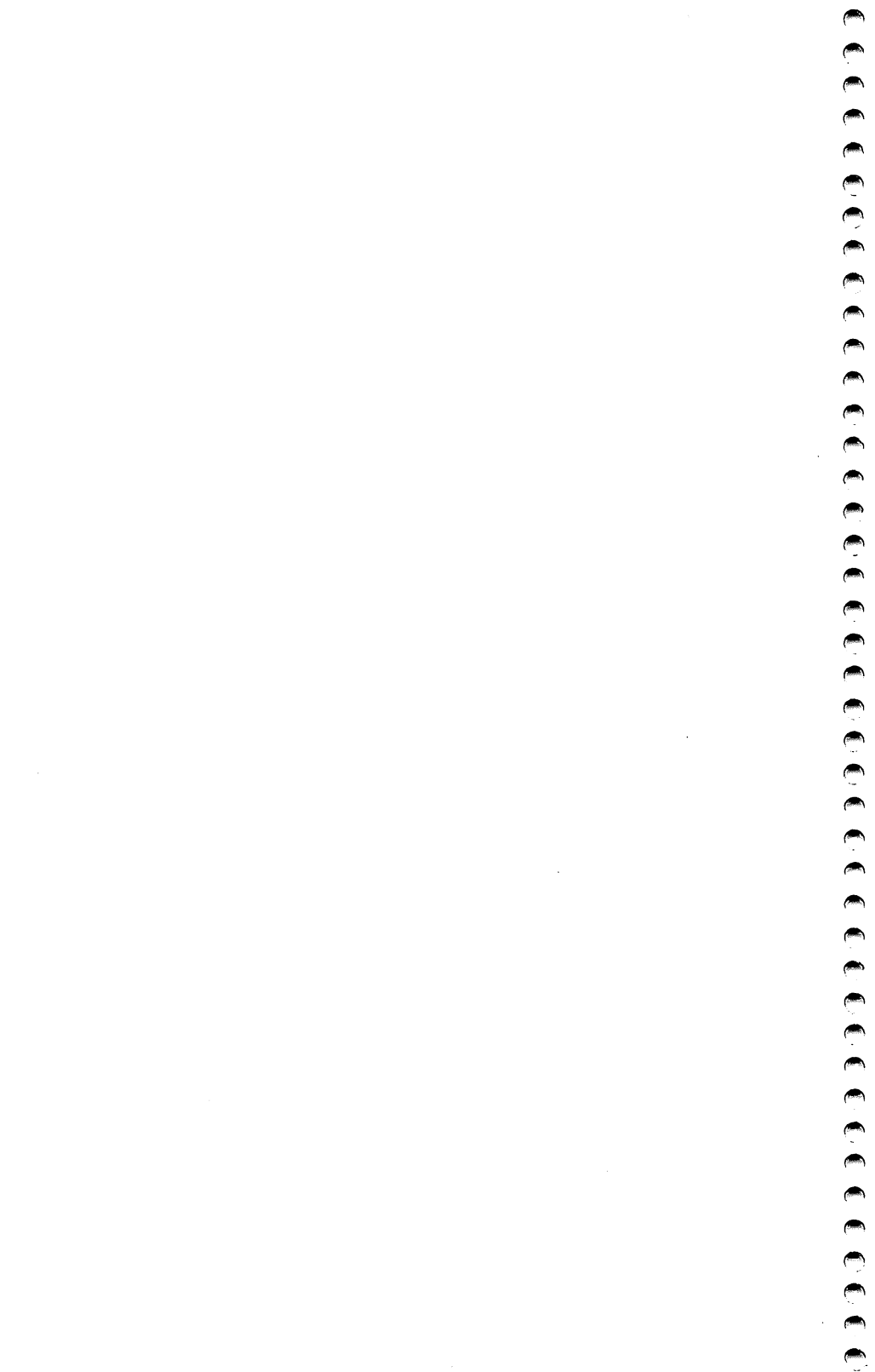
### Section 1.d Hardware Requirements

The Enhancer ][ may be installed on any Apple ][ or Apple ][ plus with a piggyback style keyboard (normally found only on revision 7 or greater Apples). Revision 0 through 6 Apples (with piggyback keyboards) can use the Enhancer ][ but will require a device for displaying lower case letters not supplied with the Enhancer ][, such as the Videoterm 80 column card.

To determine the revision of your Apple, look inside along the left hand edge of your motherboard. You should see white letters next to each row of chips. If next to the letter E you see a socket labeled "memory select", you have a revision 0 through 6 Apple. Otherwise, you should see a white dashed rectangle with three large holes. This is a revision 7 or greater Apple.

## Chapter One: Introduction

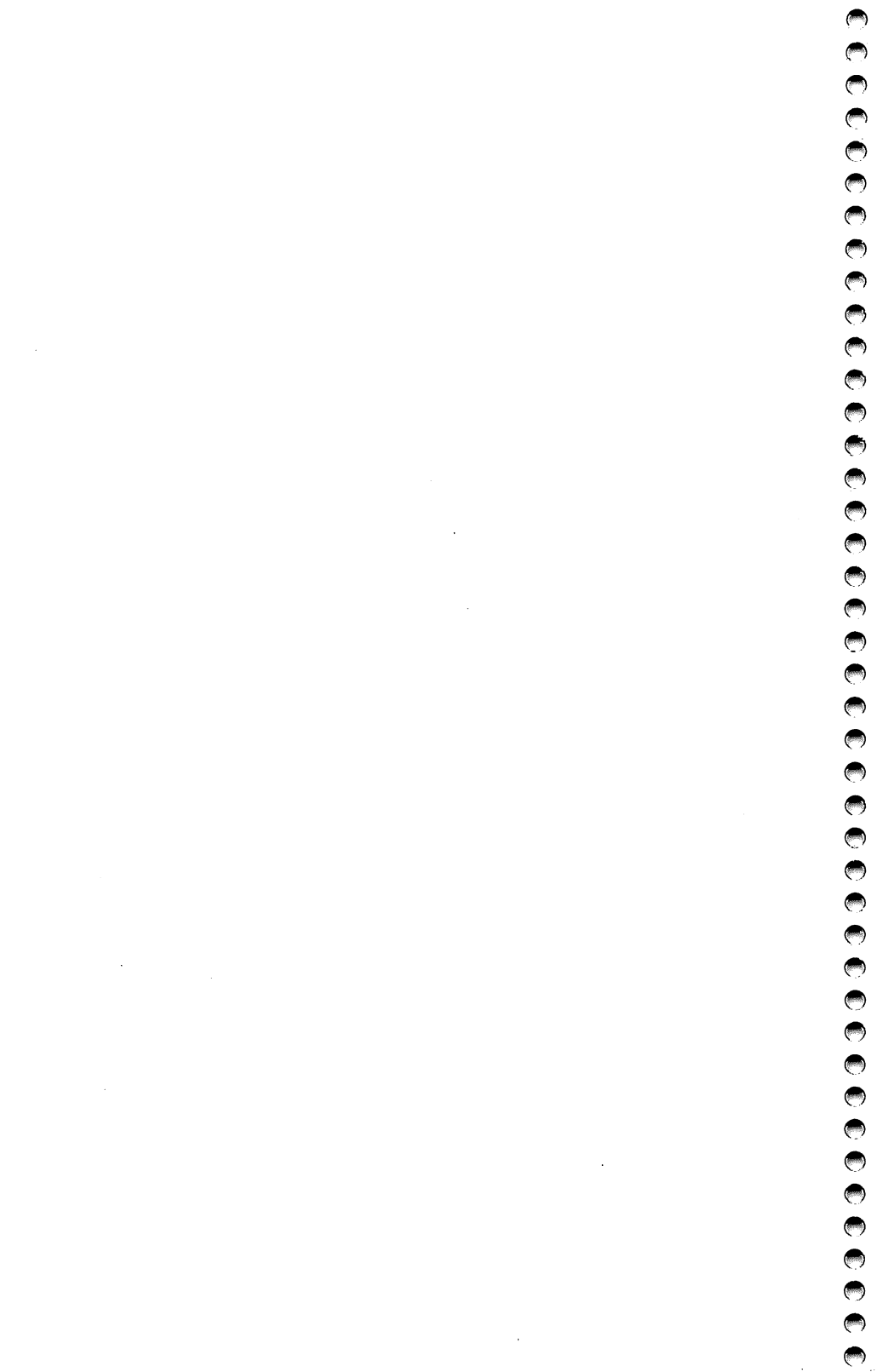
Note: This manual assumes that the user has a revision 7 or greater Apple ][. If you have a revision 0 through 6 Apple, you should NOT install the Lower Case Chip in your system. All references to lower case display (in 40 columns) would, therefore, be invalid for your system unless you have some kind of 40 column lower case display device [appendix B].



## Chapter Two: Getting Started

### Chapter Table of Contents.

- 2.a) Enhancer ][ Utilities Disc Checkout
  - 2.a.1) The BASICS Side
  - 2.a.2) The Pascal Side
- 2.b) How to Remove the Case From Your Apple
- 2.c) Lower Case Chip Installation and Checkout
  - 2.c.1) Tools Required
  - 2.c.2) Procedure
- 2.d) Enhancer ][ Installation
  - 2.d.1) Tools Required
  - 2.d.2) Procedure
    - 2.d.2.1) Installation of The Acknowledge Wire
    - 2.d.2.2) Enhancer ][ Installation
    - 2.d.2.3) Automatic Download Wire
- 2.e) Enhancer ][ Installation Checkout





## Chapter Two: Getting Started

### Section 2.a Enhancer ][ Utilities Disc Checkout

Since the disc which comes with the Enhancer ][ is not copy protected, the user should ensure that the disc they receive is in good working order and that adequate backups are made by the user to insure against data loss.

The Enhancer ][ Utilities Disc is a double sided disc. The primary (label) side is the BASICs side. The second (back) side is the Pascal side. Since the Pascal side is intended to be used only once, damage due to double sided usage is unlikely. The Pascal side is not notched and is therefore write protected.

#### Section 2.a.1 The BASICs Side

To verify the data on the BASICs side, boot DOS, insert the Enhancer ][ Utilities Disc into your drive and type the following:

```
VERIFY CHECK OUT
EXEC CHECK OUT
```

If I/O ERROR appears anywhere, you may have a problem. Repeat the steps, if an error still continues, you will need to get a new copy of the disc from your dealer.

This concludes the checkout of the BASICs side of the Enhancer ][ Utilities Disc. If you have a Pascal system, continue with the next section, otherwise the disc checkout is now complete.

#### Section 2.a.2 The Pascal Side

If you do not have the Pascal language system, you should ignore this section. If you do have Pascal, you will want to verify the Pascal side of the utilities disc. To do this, perform the following:

```
Boot Pascal.
To enter the Filer type: F
```

Once in the Filer, place the Enhancer ][ Utilities Disc into one of your drives. To do this, hold the disc with label as you normally would, then turn it over by rotating your wrist. This should result in the correct orientation of the disc. To verify, the label should be on the bottom of the disc as you are ready to insert it and it should be the last part of the disc which will enter the drive when you insert the disc.

## Chapter Two: Getting Started

```
Type: B
Pascal will ask: "Bad block scan of what vol?"
Type: Enh2:
Pascal will ask: "Scan for 28 blocks ? (Y/N)"
Type: y
```

If it says "0 bad blocks", your disc is probably good, otherwise it will name any files which are in danger. If no files are in danger, don't worry, even if there are many bad blocks found. If the files OUTPATCH.CODE and OUTPATCH.TEXT are both listed as endangered, then you will have to get at least one good copy of these files. You can ask your dealer or call Videx.

### Section 2.b How to Remove the Case From Your Apple

- ( ) Ensure that your Apple ][ is functioning properly.
- ( ) Turn your Apple ]['s power off!
- ( ) Disconnect the power cord from the power supply.
- ( ) Now, remove the lid from your Apple. Curl your fingers under the back corners of the lid, bracing your hands against either side of it. Pull it up until it pops loose. Do NOT pull the lid straight up, slide it back until its front edge clears the keyboard end of the case, then lift it clear. This procedure avoids prying your keyboard off its mounts.
- ( ) Remove all peripheral cards from slots 0-7. This may be done by rocking them forward and back while pulling up, until they come free. If you have a card in slot 0 (far left), it may be necessary to remove a short cable leading from it to the motherboard.
- ( ) If you have an RF modulator connected to the four-prong video output connector of the Apple ]['s motherboard, disconnect it.
- ( ) Turn your Apple ][ upside down.
- ( ) Remove the screws around the edge of the Apple ][ which hold the case on [photo: 2.1]. The four screws at the

Chapter Two: Getting Started

Photo: 2.1

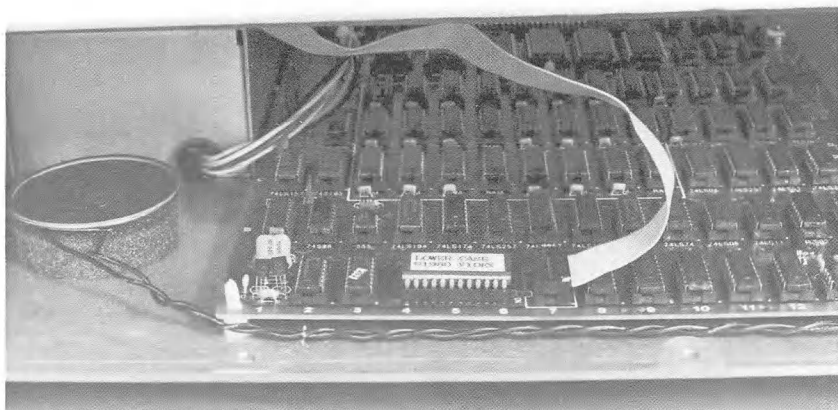


Photo: 2.2

## Chapter Two: Getting Started

front edge, under the keyboard, may have washers mounted BETWEEN the Apple ][ case and the bottom. When you remove these screws, lift the keyboard end of the base off the case and remove these washers.

- ( ) Holding the top and bottom together, carefully turn the Apple ][ over (top side up).
- ( ) Gently lift the front of the case just enough to reach in and pry loose the keyboard's ribbon cable connector [photo: 2.2] using a screwdriver to lift first one end then the other. If you plan to use your hand be prepared to have two holes in your thumb and two bent pins on the connector. NOTE CAREFULLY the orientation of the cable; you will have to put it back the same way later.
- ( ) Completely remove the case from the bottom of the Apple.

### Section 2.c Lower Case Chip Installation and Checkout

The Lower Case Chip replaces the character generator chip (A5) on your motherboard. No special knowledge of computer systems nor electronics is required. No soldering nor cutting of traces necessary. You should be able to do it yourself without any special tools within a half hour. Please read this section thoroughly before attempting to install your new Lower Case Chip.

#### Section 2.c.1 Tools Required

These are the tools required to install your Lower Case Chip:

- i) Phillips screwdriver
- ii) Standard blade screwdriver or IC extractor

#### Section 2.c.2 Procedure

Please read these instructions carefully and completely before attempting to install your Lower Case Chip. If you have not already done so, refer to Section 2.b for instructions on removing the case of your Apple.

- ( ) Locate and remove the large chip at socket location A5 on the motherboard [photo: 2.4]. Wrap it in tinfoil and set it aside where it will not be lost.
- ( ) Place the chip labeled Lower Case in the A5 socket on the motherboard. The notched end of the chip should be to the left.

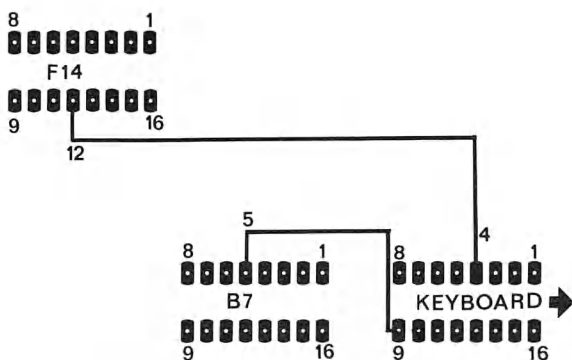


Photo: 2.3

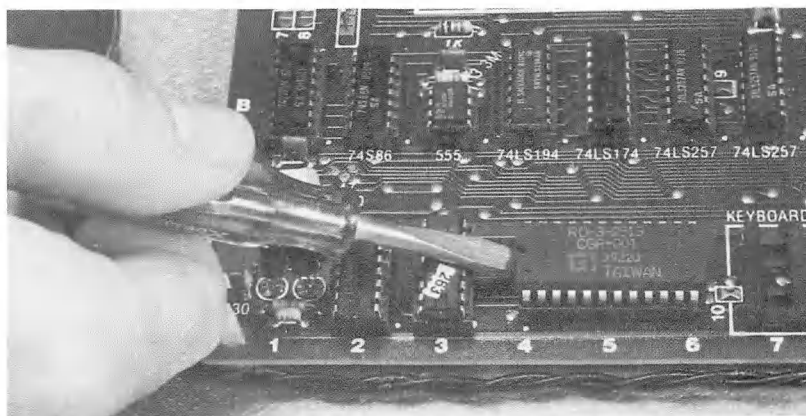
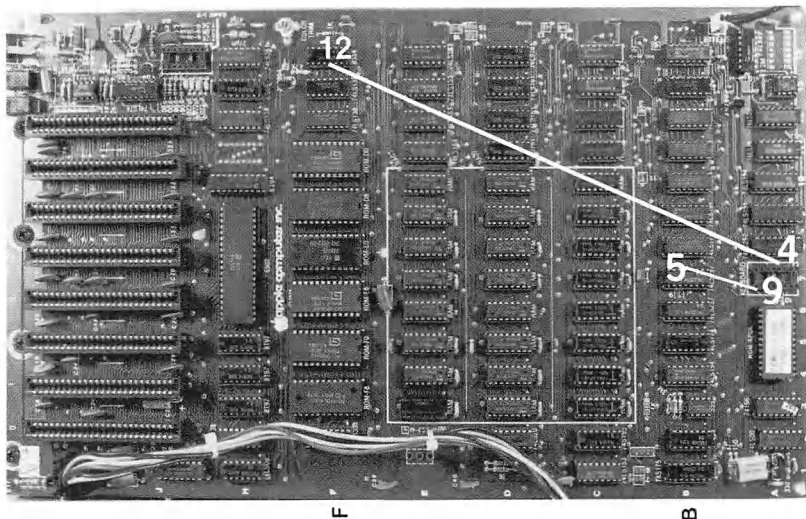


Photo: 2.4

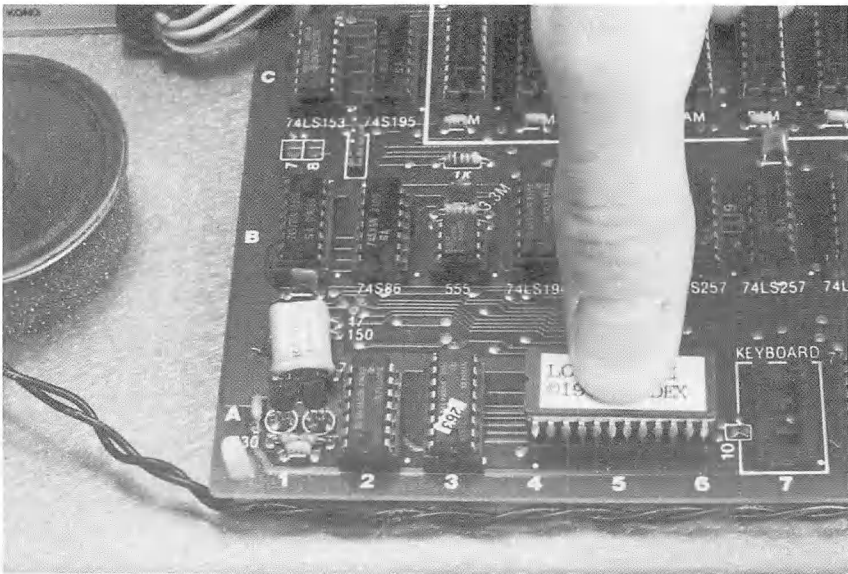


Photo: 2.5

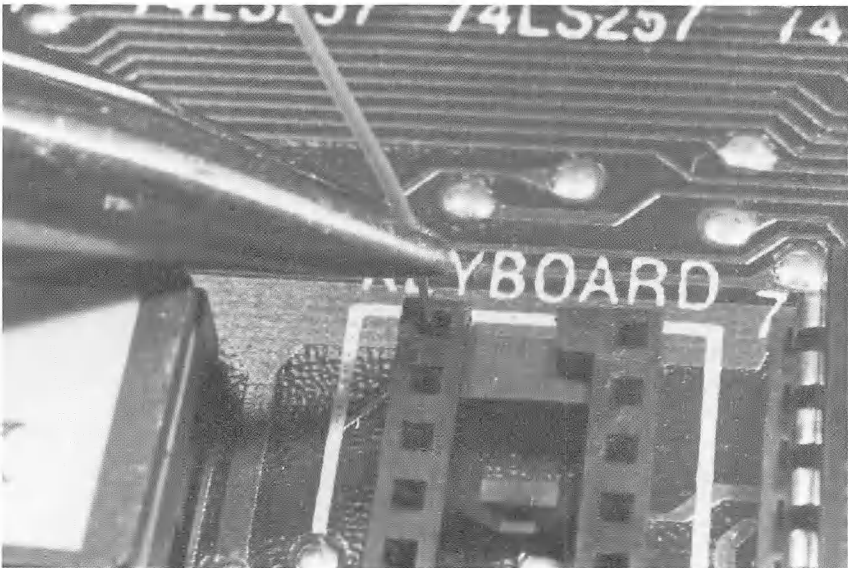


Photo: 2.6



## Chapter Two: Getting Started

- ( ) Connect your monitor or RF modulator to the Apple's video output.
- ( ) Connect your power cord to the Apple's power supply.
- ( ) Turn your monitor or television set on.
- ( ) Turn the power on to your Apple ][ and listen for its initial beep. If the power light does not come on, turn the power switch off and check the power connections. If the power light does come on, but you do not hear a beep, carefully recheck all your installation.
- ( ) If you see vertical lines, graphic characters, or anything unusual on your screen, your Lower Case Chip is probably not installed correctly. Check to see that the notched end is to the left. Carefully check to see if any pins might be bent. If the problem does not become obvious, refer to Section 3.b.
- ( ) If you see the normal characters on your screen, your Lower Case Chip is probably installed correctly.
- ( ) Proceed to the Enhancer ][ installation section [section: 2.d]

End of the Lower Case Chip installation.

### Section 2.d Enhancer ][ Installation

The Enhancer ][ replaces the encoder board of your keyboard, making installation relatively simple. No special knowledge of computer systems nor electronics is required. There is no soldering nor cutting of traces necessary. You should be able to do it yourself without any special tools within an hour. Please read this section thoroughly before attempting to install your new Enhancer ][.

#### Section 2.d.1 Tools Required

These are the tools required to install your Enhancer ][:

- i) Phillips screwdriver
- ii) Pliers (preferably needle-nose)
- iii) Wire cutters (optional)
- iiii) Wire stripper (optional)

## Chapter Two: Getting Started

### Section 2.d.2 Procedure

Please read these instructions carefully and completely before attempting to install your Enhancer ][. If you have not already done so, refer to Section 2.b for instructions on preparing your Apple.

#### Section 2.d.2.1 Installation of The Acknowledge Line

This section describes the installation of the Acknowledge wire (this enables use of the type ahead buffer and ensures proper operation of the macro definitions). While this wire is not necessary, it is strongly encouraged.

- ( ) Cut a length of wire about two inches long (about 5 cm).
- ( ) Strip one eighth inch (or 3 mm) of insulation from each end.
- ( ) Locate on the Apple motherboard the integrated circuit (IC) at location B-7 [photo: 2.3]. It is in the second row from the front and seventh from the left edge of the motherboard, and is marked 74LS257 both on the IC and on the motherboard next to it.
- ( ) Remove the 74LS257 by using a flat-bladed screwdriver to pry up first one end, then the other, until the IC is free. Use needle-nose pliers to straighten any pins you may have bent.
- ( ) Locate pin number 5 of the empty socket at B-7 [photo: 2.7]. Insert one of the stripped ends of the short wire into this hole.
- ( ) Examine the 74LS257 again for bent pins, then insert it into the socket, letting pin 5 join the wire in the hole. BE SURE that the end of the IC with the notch or dimple is pointed toward the keyboard socket [photo: 2.7]. If it is installed backward it and/or your Apple could be damaged.
- ( ) After inserting the IC in B-7, check that the bare part of the wire does not touch any pins other than pin 5.
- ( ) Bring the other end of the wire over to pin 9 of the keyboard cable socket at location A-7. Insert it into pin 9, making sure that it will not touch any other pins of the keyboard cable plug.

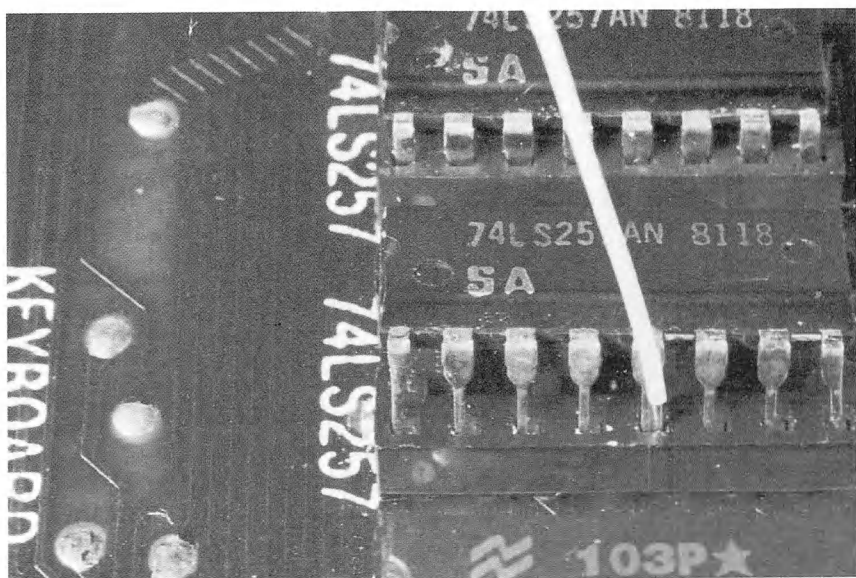


Photo: 2.7

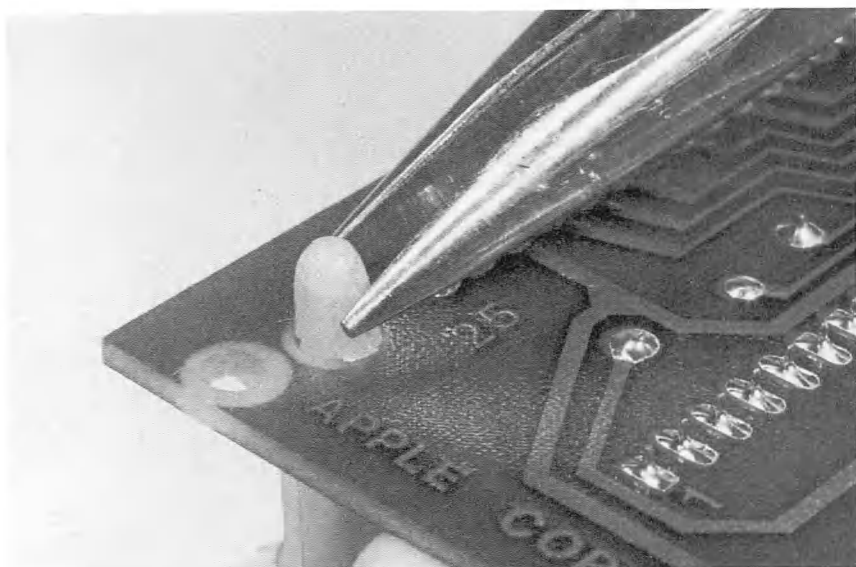


Photo: 2.8

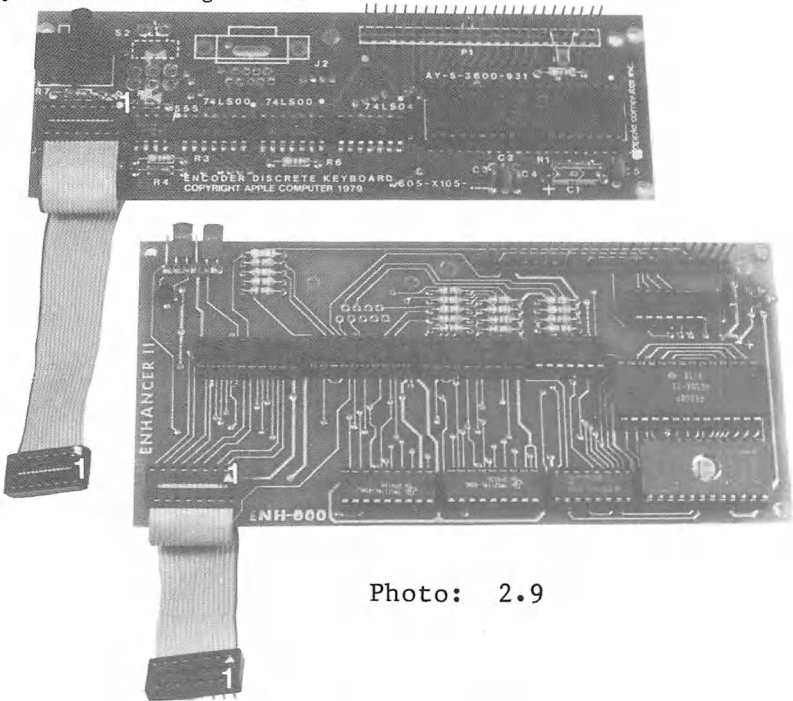


Photo: 2.9

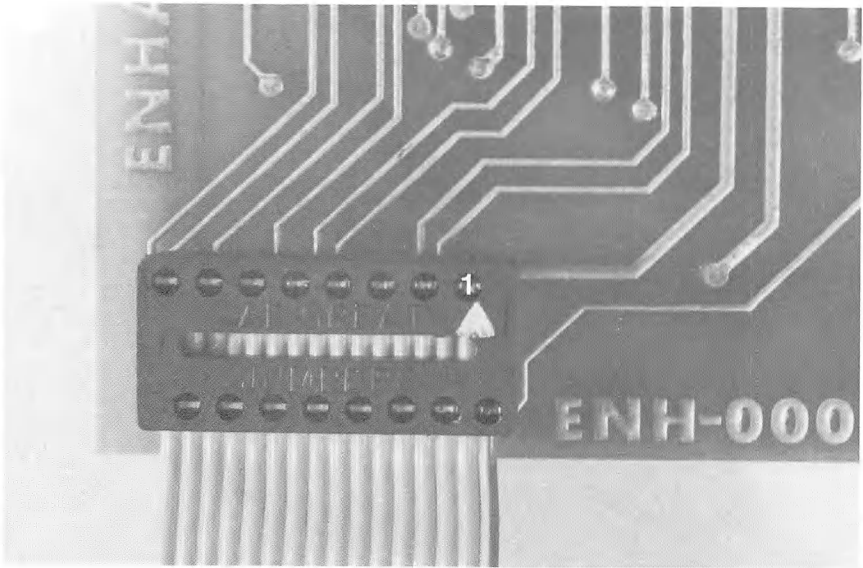


Photo: 2.10

## Chapter Two: Getting Started

### Section 2.d.2.2 Enhancer ][ Installation

- ( ) Invert the case of your Apple and observe the keyboard and the "piggyback" encoder board mounted on it.
- ( ) Note the nylon spacers that extend through the piggyback board. The board will have to slide off of these spacers. Squeeze the ends of the spacers with your pliers so they will fit through the board's holes [photo: 2.8]. At the same time, pull the piggyback board away from the keyboard. There will be some resistance from the 25-pin connector (the contacts are spring-loaded) but it should slide smoothly out, once the spacers are freed.
- ( ) Very carefully remove the 16-pin cable connector from your piggyback board, using the flat screwdriver to pry up alternately one end, then the other, until it is free. Note the orientation of the connector. You will have to plug it into your Enhancer ][ from the same direction [photo: 2.9].
- ( ) Examine the 16 pin cable. At each end you should see a white triangle or dot or perhaps some numbers. The dot or other mark indicates pin one.
- ( ) Examine the Enhancer ][ circuit board. The socket at the lower left hand corner is the keyboard cable socket. The upper right hand pin is pin number one. Take the 16 pin cable and insert it into this socket, taking care to ensure that pin one of the cable is inserted into pin one of the socket [photo: 2.10].
- ( ) Examine the 25-pin connector on the Enhancer ][. Make sure all pins are straight and parallel. Straighten any that are not.
- ( ) Now, look at the underside of your keyboard. On some Apples there will be a metal stiffener bar across the back of the keyboard, extending up about three-eighths inch from the keyboard surface. If this is present, it must be covered before your Enhancer ][ is installed further. There should have been included in your Enhancer ][ box a strip of insulating material about four inches long. This should be placed over the edge of the bar so that when the Enhancer ][ is installed, no bare metal touches it [photo: 2.12].
- ( ) Look now at the plastic spacers that held the piggyback board on. There will be a center post flanked by two curved flanges. The right spacer (the one further from the side of the Apple) must be rotated 90 degrees so the flanges are parallel to the edge of the keyboard, as shown in photo 2.13.

Chapter Two: Getting Started

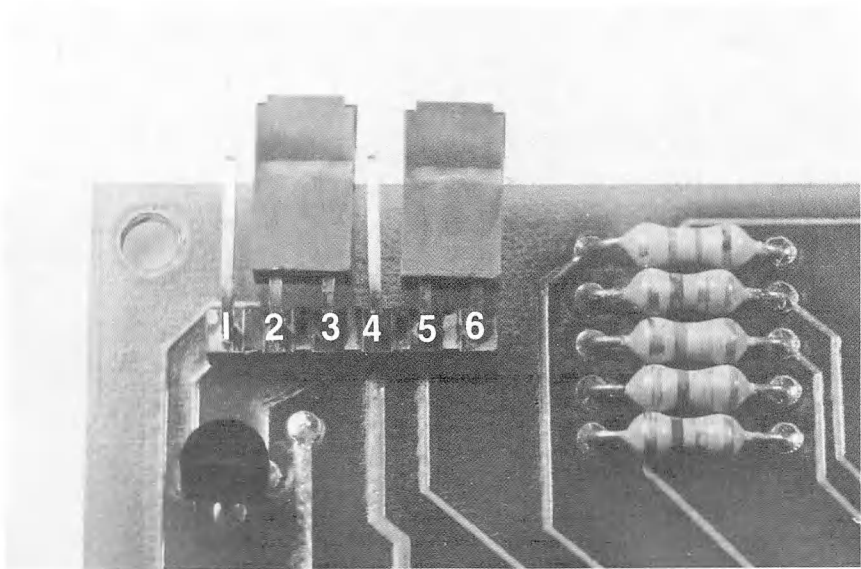


Photo: 2.11

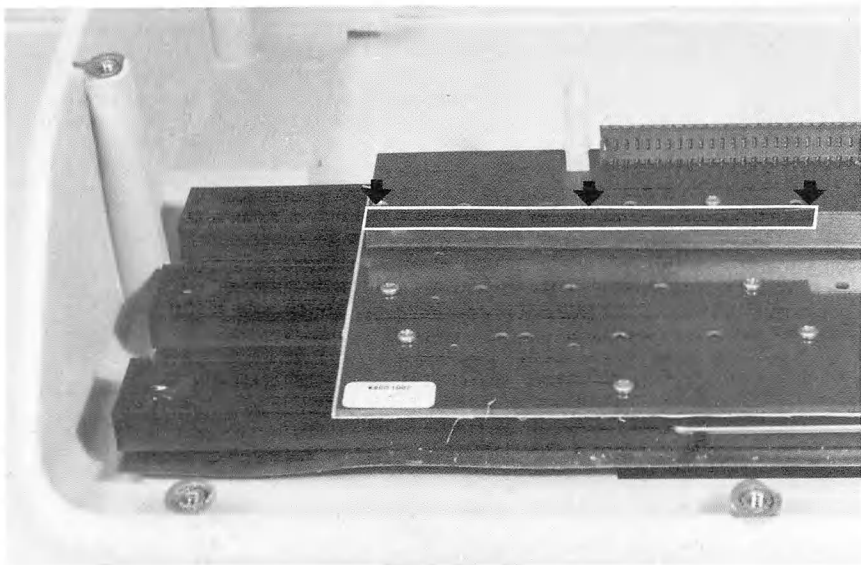


Photo: 2.12



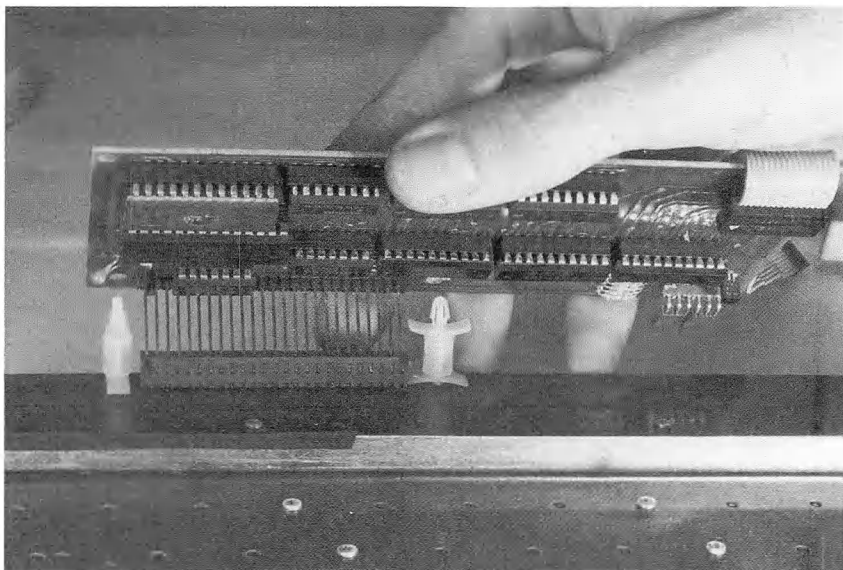


Photo: 2.13

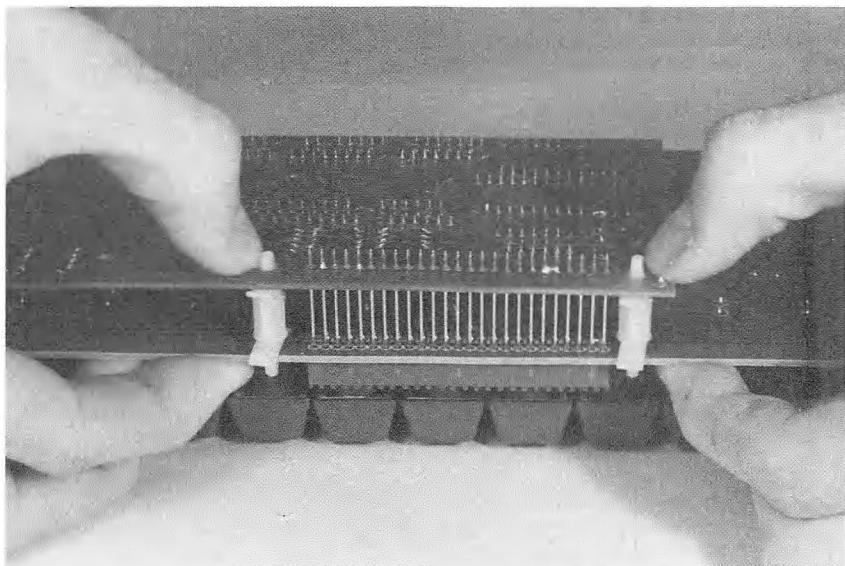


Photo: 2.14

## Chapter Two: Getting Started

- ( ) Once the 16-pin ribbon cable is installed and any of the 25 pins straightened as necessary, you may install the Enhancer ][ on the keyboard in place of the piggyback board you removed. To line up the pins, tilt the board a little so the tips of the pins press on the edges of the holes in the keyboard [photo: 2.13]. By wiggling the Enhancer ][ you should be able to make most of the pins pop into the correct holes, any that do not may be lined up with a screwdriver.
- ( ) When all the pins line up, press the Enhancer ][ smoothly onto the keyboard until the spacers lock in place.

### Section 2.d.2.3 Automatic Download Wire

This section concerns the installation of a wire that allows totally automatic downloading of key redefinitions. If you are not sure you wish to have this, refer to Section 5.f.4 for an explanation of its effects.

**IMPORTANT:** If you install this wire, the blue plastic cap on Molex pin five [photo: 2.11] must be removed.

- ( ) Cut a length of wire about twelve inches (30 cm) long.
- ( ) Strip one eighth inch (or 3 mm) of insulation from each end.
- ( ) Locate on the Apple motherboard the integrated circuit (IC) at location F-14. It is in the sixth row from the front, at the right edge of the motherboard, and is marked as a 9334 (or 74LS259) both on the IC and on the motherboard next to it. NOTE: There may be a Soft Video Switch installed in this socket. If so, install the wire under the SVS, not the 9334.
- ( ) Remove the 9334 by using a flat-bladed screwdriver to pry up first one end, then the other, until the IC is free. Use needle-nose pliers to straighten any pins you may have bent.
- ( ) Locate pin number 12 of the empty socket at F-14. Insert one of the stripped ends of the long wire into this hole.

## Chapter Two: Getting Started

- ( ) Examine the 9334 again for bent pins, then insert it into the socket, letting pin 12 join the wire in the hole. BE SURE that the end of the IC with the notch or dimple is pointed toward the keyboard. If it is installed backward it will be destroyed when you turn the power on.
- ( ) After inserting the IC in F-14, check that the bare part of the wire does not touch any pins other than pin 12.
- ( ) Bring the other end of the wire over to pin 4 of the keyboard cable socket at location A-7. Insert it into pin 4, making sure that it will not touch any other pins of the keyboard cable plug.
- ( ) Make sure the blue shorting plug in photo 2.11 is removed.

End of download-control-wire installation

- ( ) Now, place the case of your Apple back on its base, left edge first. Be careful not to crush the keyboard cable connector.
- ( ) Lift the right side of the case just enough to reach in and grab the keyboard cable connector plug. Plug the connector into its socket, making sure that pin one of the cable enters hole number one of the socket (this hole will have a white dot silkscreened next to it).
- ( ) Reinstall two of the screws immediately under the keyboard, but don't bother replacing the washers. (If your Apple is on a table you should be able to do this without inverting it.)
- ( ) Proceed to the installation checkout section. If everything checks out ok, complete the re-assembly of your Apple ][ (with the power off), else refer to Chapter 3.

End of installation.

## Chapter Two: Getting Started

### Section 2.e Enhancer ][ Installation Checkout

The purpose of this section is to determine whether your Enhancer ][ is installed and functioning correctly. If these steps are followed precisely, independent variables which can give the appearance of disfunction will be controlled. If you are unable to perform any of these steps, proceed to Chapter three.

- ( ) Turn the power off (important).
- ( ) Disconnect any and all peripheral cards which may be installed in slots 0-7.
- ( ) If your monitor is not connected to the Apple's video output, connect it.
- ( ) Turn the power on.
- ( ) If the power light does not come on, shut off the power immediately and check your connections. Probably your keyboard cable is plugged in backwards, this can damage your Enhancer ][. If it does not become obvious, turn to Chapter 3: Troubleshooting.
- ( ) If the power light does come on but there is no beep and no prompt on the screen, check your connections, and check that the IC's you had pulled are plugged in right. Again if it's not something obvious, turn to Chapter 3.
- ( ) If the you get a beep but no prompt on the screen, check that your monitor is connected correctly and powered up.
- ( ) If you get a beep and a prompt, you are ready to begin checking your Enhancer ][. If a message is displayed, either ENHANCER CHECKSUM FAILURE, ENHANCER MEMORY FAILURE, BREAK ERROR, or INTERRUPT, your Enhancer ][ has failed its own self-check and must be fixed. Refer to Section 3.a.
- ( ) Type some random characters, just to see that you can enter characters. If none appear, your Enhancer ][ is not working yet, recheck your installation, then refer to Chapter 3 if you do not find anything.

## Chapter Two: Getting Started

- ( ) If characters are being entered, you may begin the full check. Enter the following keys several times each, and check that they are echoed to the screen properly.

Q W E R T Y U I O P then 3 Q S D Z then

@ A B C D E F G H I J K L M N O P Z

<ctrl>G (expect a beep)

<ESC> then I (expect the cursor to move up one line)

(<ctrl> and <ESC> are the keys with those labels on them).

The first set of characters composes one row and one column of the keyboard matrix, Table 3.1. The second defines a row and column of the ASCII chart, Table 5.1 [section: 5.b.1]. Problems tend to show as patterns of errors associated with the arrangement of one of these charts. If these keys all produced the correct characters, skip the next three steps.

- ( ) If any of the keys you pressed did not echo as the correct character, try each of the other keys in that row and column. The results of this check should be noted down for use in servicing your Enhancer ][.
- ( ) If the correct character is being entered but is followed by an underline character, then PROBABLY your download-control wire, the one leading from the keyboard socket to position F-14 near the back edge of your motherboard is not connected well. To check this, remove the wire and reinstall the blue shorting plug across pins 5 and 6 of the Molex connector, then repeat the checkout sequence.
- ( ) If the wrong characters are entered but do not show a regular pattern on Table 3.1, check the characters against the chart in Chapter 5, Table 5.1. Again, if you can detect any pattern, note it down.
- ( ) Assuming the characters tried so far have been correct, we can now check the rest of the Enhancer ][. Begin by pressing RESET by itself. It should not cause a system reset (a beep), if it does, the (left) blue shorting plug to the left in photo 2.11 is in the wrong position, or something is not right with your Enhancer ][.
- ( ) Enter a `G`, it should be displayed as such.
- ( ) Holding CTRL down, enter a `G` again, you should hear a beep.

## Chapter Two: Getting Started

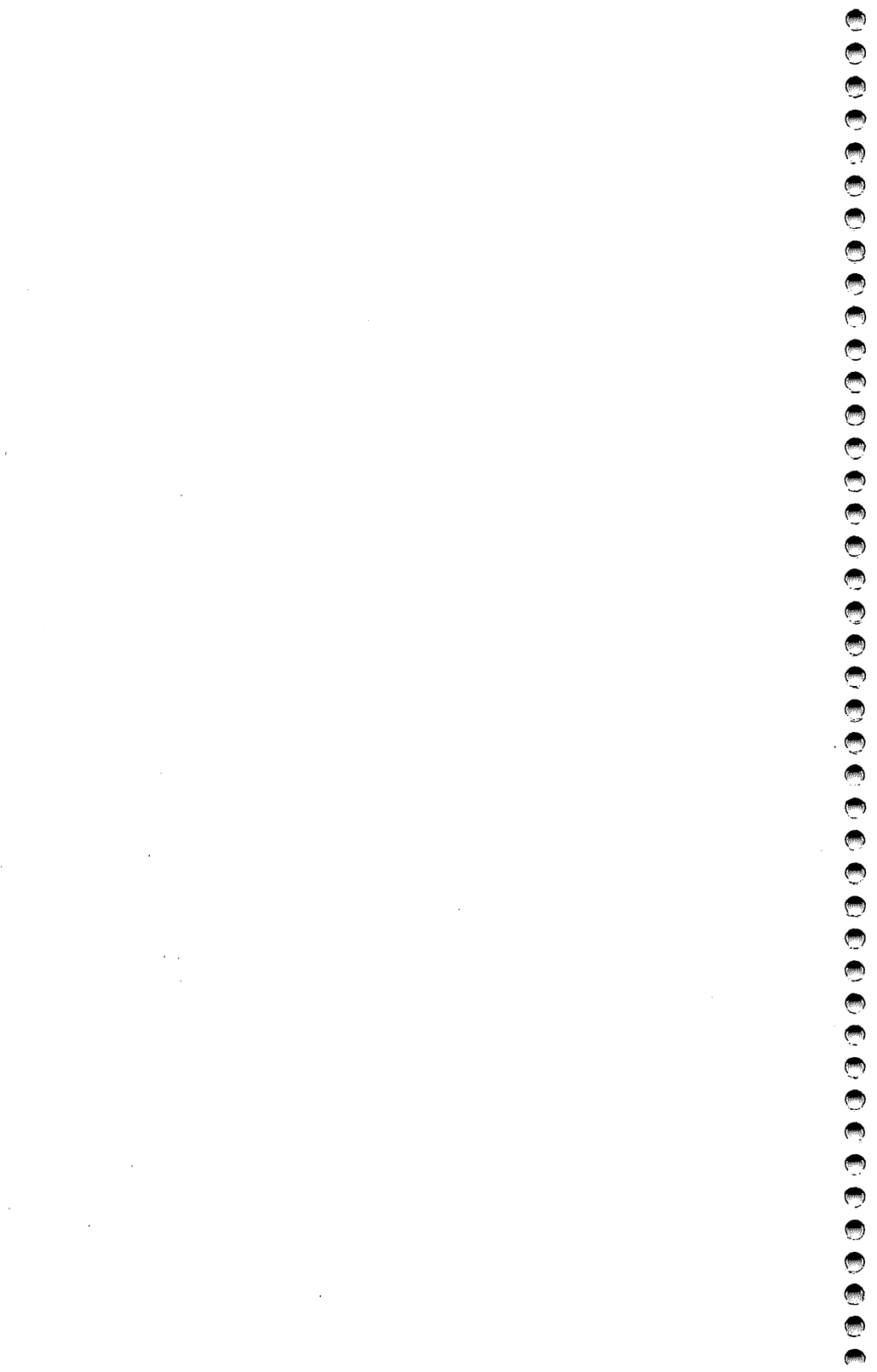
- ( ) Enter an 'N', you should see an 'N' displayed.
- ( ) Now, hold SHIFT down while entering an 'N', you should see an '^' displayed.
- ( ) If these characters work OK, then your character keys, your control key, and your shift key are known to work. Now check your REPEAT function. Hold REPEAT down and press another key such as 'P'. It should IMMEDIATELY begin repeating at high speed.
- ( ) If you installed the wire that enables the type ahead buffer, check it this way: Fill a line with characters, then hold down REPT and RETURN for a few moments, then let them up, the screen should continue scrolling for a moment afterwards. If it does not, check installation of the type ahead wire.
- ( ) Now, hold down the shift key, press reset and let it up, then let up the shift key. This puts your Enhancer ][ in Lower Case mode.
- ( ) Now enter an 'N' then a <shift>'N', both should result in an upper-case 'N'. (The Enhancer ][ is actually producing a lower case 'n', but the Apple is converting it to upper case.)
- ( ) The following steps check the automatic download capability. If you installed the wire that enables this, check it this way: If the prompt now displayed at the left side of your screen is a ']' or a '>', proceed to the next step, otherwise, enter a CTRL-'B' then RETURN.
- ( ) Now, assuming you have a BASIC prompt, enter the line:  
  
    <cntl-reset> ~~PRINT~~ POKE -16290,0  
  
    and press RETURN. There should be an underline character displayed just to the left of the cursor. If not, the download control wire is not properly installed.
- ( ) This concludes manual checkout of the Enhancer ][. You may wish to try the rest of the codes in Table 5.1, especially if you had installed the keyboard cable connector backwards.

You may now proceed to Chapter 4 for an introduction to your new Enhancer ][.

## Chapter Three: H E L P ! or What To Do If All Else Fails

### Chapter Table of Contents.

- 3.a) Save Time & Money
- 3.b) Trouble-Shooting
  - 3.b.1) The Lower Case Chip
  - 3.b.2) The Enhancer ][
  - 3.b.3) Later Problems
- 3.c) The RMA Form
  - 3.c.1) How to Complete the RMA Form
- 3.d) How to Package your Enhancer ][ Before Shipping It Back for Repair





## Chapter Three: H E L P ! or What To Do If All Else Fails

### Section 3.a Save Time & Money

Careful reading of this chapter can save you both time and money. It covers most of the difficulties that you are likely to encounter. You should read this chapter in its entirety prior to calling Videx or shipping the Enhancer ] [ back for repair. You will probably want to skip the entire chapter unless you encounter some nasty snag in installation or operation.

A great deal of care and testing goes into each product that we make. Your Enhancer ] [ was exhaustively tested by our quality control department prior to being packaged. Though it may not seem like it, the problem which you have encountered is probably not due to equipment failure. By reading this chapter carefully, you will probably be able to resolve any problem yourself without costly long distance phone calls or needless shipping of a perfectly good Enhancer ] [ for repair. This, in turn, will tend to reduce "system down time." If after you have read this chapter, carefully checked the operation of the Enhancer ] [ and you are still having difficulty, please give us a call.

All products returned to Videx must be accompanied by a RMA (returned merchandise authorization) number. To receive an RMA number, you must call or write Videx.

### Section 3.b Trouble-Shooting

That you are reading this implies that something does not seem to be working right with your Enhancer ] [. The following sections are arranged in the order in which problems would be discovered.

#### Section 3.b.1 The Lower Case Chip

This section refers to problems that occur in following the instructions in Section 2.c. There are a number of problems that could occur:

- a. The power comes on, but the Apple does not beep.
  1. Check installation, probably the Lower Case Chip is installed backwards or offset one pin, this results in a destroyed Lower Case Chip.
- b. The Apple beeps, but the screen remains blank.
  1. Check that the monitor is plugged in
  2. Turned on
  3. The cable is connected
  4. The RF modulator, if any, is connected correctly.

## Chapter Three: H E L P ! or What To Do If All Else Fails

- d. The screen lights up, but no characters are displayed. There may be horizontal or vertical bars, or no real pattern at all.
  1. Check for bent pins on the Lower Case Chip.
  2. The Lower Case Chip may be faulty, replace it with the original and see if that works now.
- e. Characters are displayed, but at random (no pattern).
- f. Characters are displayed, but they are the wrong characters for those positions.

NOTE: Turning the power on with the Lower Case Chip plugged in backwards will destroy the Lower Case Chip. If this occurs, you must return the Lower Case Chip for exchange.

Returned merchandise must be accompanied by an RMA form, a detailed description of the problem will greatly speed service.

### Section 3.b.2 The Enhancer ] [

The Enhancer ] [ is a complex device, many different problems may occur, some gross, some subtle. This section begins with those problems that may occur on power-up.

1. No power light, power supply make snapping noise: power supply is shorted out. Remove the keyboard connector cable from A-7 and try again. Restore one feature of your Apple to its original state at a time, each time checking whether your Apple works afterward. If your Apple is returned to its pristine state and still does not work, there probably is a damaged chip on it, but first check that all the chips are plugged in with the notched or dimpled end pointed TOWARD the keyboard end of the A Apple, any that are plugged backwards will grow very hot immediately after powerup. If the Apple works (beeps on powerup) after removing the keyboard cable, there is something drastically wrong with your Enhancer ] [. Since each is checked before leaving Videx, the most likely cause is either: one (or both) of the wires leading to the wrong hole in a socket, or one of the blue shorting blocks on the wrong pair of Molex pins (if they should be there at all).
2. No power light, no snapping noises: Your keyboard cable is probably plugged backwards (This would damage your Enhancer ] []). If not, check other things: Assuming you had already checked your Lower Case chip with the keyboard cable disconnected, you should check that each chip you removed is plugged right, with no pins bent under or out; that the wires are plugged into the right

### Chapter Three: H E L P ! or What To Do If All Else Fails

holes, that they are not touching (and cannot touch) any other pins; check that the wire leading to your speaker is free, not pinned to the bottom of your motherboard.

3. Power light, no beep: again, follow the instructions in step 2
4. Power light, beep, no display: check the conditions in b) above, as well as those in step 2. Check the shorting plugs against photo 2.d.<> and the instructions in Section 2.d.
5. Power OK, beep, APPLE ][ display, Basic prompt; no character entry, no messages displayed: When there is anything serious wrong with the Enhancer ][, this is how it will usually fail. Is there a bar across the back of your keyboard? If so, check that it is adequately insulated from the Enhancer ][. Is there any other equipment under the position where the Enhancer ][ mounts? If so, do they touch? The bottom side of the Enhancer must not be allowed to touch bare metal. If there is anything occurring additionally, be sure to note it on the RMA form.
6. Characters being entered, but erratically: this includes any instance of characters being entered when you haven't been typing anything, usually it involves a single character repeating indefinitely. Check that the wires installed in the keyboard socket enter the correct holes, and that they do not touch any pins other than the proper ones.
7. Characters entered when key is struck, but more than one character: If it is an underline, the download-control-wire is improperly installed, or the blue plastic shorting plug is on the wrong pair of Molex pins [photo 2.d.x]. If it is a set off characters, see if there is any correspondence between which ones are produced for a certain character and keys in Table 3.1 on the RMA form. Check the repeat speed, if it is incredibly fast, write that down.
8. One character entered when one key is struck, but (sometimes) the wrong character: Again, check for correspondence against both Tables, 3.1 and 5.1. Note any patterns.

### Section 3.b.3 Later Problems

This section deals with problems that occur some time after installation checkout. It assumes that you had successfully run through the installation checkout procedure some time before.

The most common problems will be:

1. No lower case entry to BASIC: refer to Section 6.a.
2. Repeat won't work well in Pascal: This is because Pascal is too slow to pick up characters as they become available. Use the "flush buffer" command (<reset> or <shift><reset>) to stop the cursor where you want it, or anytime Pascal runs away from you.
3. Shift-lock is too easily set: this feature may be turned off by the down-load program supplied with your Enhancer ][.
4. Macros defined earlier are gone: what mode were they defined under, Caps Lock or Lower Case? What mode are you in now? Try switching to the other mode and using the macro -- it may be there! Is it possible that you have pressed <rept><reset> or shut off your Apple since defining the macro? Are you running a program that could be doing things with Annunciator #3, perhaps as a "copy-protect" scheme? If so, you may wish to use the down-load program to disable automatic downloading (this would have to be done only once after turning the power on but prior to running the offending program).
5. The Enhancer ][ works fine for about a half-hour, then quits: First, check if you haven't gotten it into a weird mode: does it think you're defining a macro? Press <space> then <rept> and try it then. If that doesn't do it, press <rept><reset>, which will clear any macros now defined, including spurious ones. If that doesn't do it, try a system reset (probably <ctrl><reset>) and try it then. If it works then, use it for another ten minutes, very suspiciously: it may be a subtle thermal problem. Finally, try turning the system off, then on again. Again, try it for awhile, with a wary eye on what it is (and you are) doing. Anything you notice should be written on the RMA form.

## Chapter Three: H E L P ! or What To Do If All Else Fails

### Section 3.c The RMA Form

All products returned to Videx for repair or replacement should be accompanied by an RMA (returned merchandise authorization) form. By completing an RMA form in detail and returning it with your shipment, you will probably cut the repair time at least by half. The RMA form ensures accurate handling and quick diagnosis of your board. You will find one of these forms in the back of this manual and another in figure 3.3. Figure 3.1 is a sample of a completed RMA form.

If your blank RMA form is missing, you may use a photocopy of figure 3.3. It is suggested that you do not write on figure 3.3 itself.

#### Section 3.c.1 How to Complete the RMA Form

Figure 3.2 is a sample of a completed RMA form. You should ensure that the RMA form is completely filled in. If the form is completed in detail, the repair time can be significantly reduced.

The RMA form is divided into four parts. Section A is for general information. Section B deals with your system's configuration. This data is useful in determining if some part of your system may be conflicting with the Enhancer ][ or related problems. Section C asks questions which we have found to be useful in determining certain problems. Section D is reserved for a complete description of the problem.

Please refer to figure 3.2 for the following discussion.

Chapter Three: H E L P ! or What To Do If All Else Fails

RMA Form for Enhancer ][ ENH-000

RMA # \_\_\_\_\_

Serial # \_\_\_\_\_

Previous Service RMA # \_\_\_\_\_

Name \_\_\_\_\_

Shipping Address:

Organization \_\_\_\_\_

Name \_\_\_\_\_

Addr. \_\_\_\_\_

Addr. \_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

Shipping Instr. \_\_\_\_\_

Phone # \_\_\_\_\_

times \_\_\_\_\_

Time Zone \_\_\_\_\_

Phone # \_\_\_\_\_

times \_\_\_\_\_

Dates purchased \_\_\_\_\_

sent \_\_\_\_\_

System Configuration:

\_\_\_\_\_ Autostart

\_\_\_\_\_ Old Monitor ROM

\_\_\_\_\_ Apple ][ plus

\_\_\_\_\_ Apple ][

Resident Language:

\_\_\_\_\_ AppleSoft

\_\_\_\_\_ Integer

Number of disc drives:

List on the back of the page all products installed in the Apple at the time the failure occurred, and any software that was in use.

For problems that occurred during installation, did you get a:

\_\_\_\_\_ power light?

\_\_\_\_\_ power-up beep?

\_\_\_\_\_ Display?

Were there any installation errors?

Any messages displayed on power-up or later?:

\_\_\_\_\_ ENHANCER RAM FAILURE

\_\_\_\_\_ ENHANCER CHECKSUM FAILURE

\_\_\_\_\_ BREAK ERROR

\_\_\_\_\_ INTERRUPT

\_\_\_\_\_ Single character repeating continuously ? (no keys pressed)

\_\_\_\_\_ Some keys produce several characters ?

\_\_\_\_\_ Some keys produce nothing ?

\_\_\_\_\_ Some keys produce Wrong characters ?

\_\_\_\_\_ All keys produce nothing ?

No effect from:

\_\_\_\_\_ Shift key

\_\_\_\_\_ Control key

\_\_\_\_\_ Repeat key

\_\_\_\_\_ typeahead doesn't work

\_\_\_\_\_ autodownload doesn't work

Does the problem occur only several minutes after powerup ?

Describe in detail, the circumstances under which the problem occurred.

# Chapter Three: H E L P ! or What To Do If All Else Fails

| rows | columns |    |    |    |    |    |    |    |    |    | legend:       |
|------|---------|----|----|----|----|----|----|----|----|----|---------------|
|      | C9      | C8 | C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |               |
| R1   | 3       | 4  | 5  | 6  | 7  | 8  | 9  | 0  | :  | -  | ~H: backarrow |
| R2   | Q       | W  | E  | R  | T  | Y  | U  | I  | O  | P  | ~U: forearrow |
| R3   | D       | F  | G  | H  | J  | K  | L  | ;  | ~H | ~U | ~]: ESC       |
| R4   | Z       | X  | C  | V  | B  | N  | M  | ,  | .  | /  | ~M: RETURN    |
| R5   | S       | 2  | 1  | ~] | A  | sp |    |    |    | ~M | sp: Space     |

Table 3.1: Substitution Chart (may be photocopied to speed service; fill in the character produced for that key)

Use tear-card at rear of manual in lieu of writing on these pages.

## Chapter Three: H E L P ! or What To Do If All Else Fails

### Section 3.d How to Package your Enhancer ][ Before Shipping it Back for Repair

Before you return your Enhancer ][ to Videx for repair, please be sure that you have carefully followed the troubleshooting section of this chapter [section: 3.b]. Many problems can be corrected by carefully reading this manual without the expense of costly long distance phone calls or needless shipping.

All returned merchandise must be accompanied by a RMA [section: 3.c] form AND a RMA number. You must call Videx for the RMA number. Enclosing a completed RMA form will greatly reduce the time necessary for repair.

Do NOT return manuals, cables, etc. Only return the board itself. This will decrease the total shipping weight and speed up our processing of your repair. Be sure to ship the boards using the boxes in which they were originally shipped, including any packing materials. Please ensure that any protruding parts on the board are adequately protected.

You should insure your shipment for replacement cost. Videx cannot assume liability for materials lost or damaged in transit.

Send all repairs to the following address:

Videx, Inc.  
Service Department  
897 NW Grant Avenue  
Corvallis, Oregon 97330

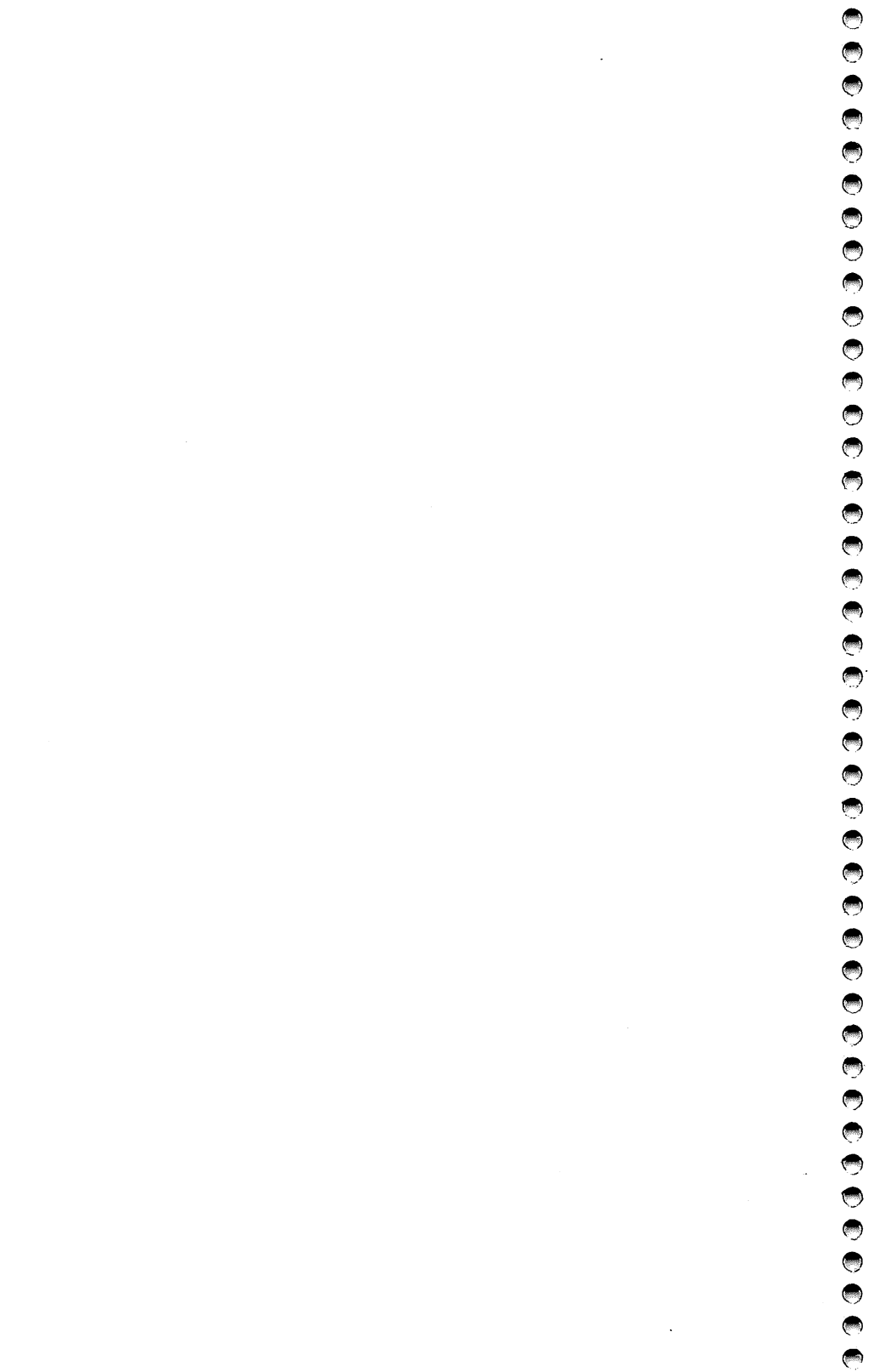
For a RMA number, call: (503) 758-0521.



## Chapter Four: A Primer for Beginners

### Chapter Table of Contents.

- 4.a) Introduction
- 4.b) Limitations of the Apple ][ Keyboard
- 4.c) What the Enhancer ][ Can Do



## Chapter Four: A Primer for Beginners

### Section 4.a Introduction

This chapter is a primer for people who do not consider themselves computer wizards. It attempts to answer many of the questions which are probably racing through your mind. It cannot anticipate all questions, however. Your Apple ][ computer is a complex piece of machinery. The Enhancer ][ makes it even more complex yet it can actually make your Apple ][ easier to use. As with any device, machines cannot be utilized to their fullest potential unless they are thoroughly understood by the operator. In the end, your best teacher will be yourself. This manual should be thought of as a tool in teaching yourself how to use your improved keyboard. It is strongly recommended that you install your Enhancer ][, if you have not already done so, before reading further so that you can try the things that you will be reading about. This will reinforce your learning and make the reading easier.

Even if you have never had any experience with computers before, you will probably be able to install and learn how to operate the Enhancer ][ in a relatively short period of time. Familiarity with your Apple ][ is a prerequisite to the use of the Enhancer ]['s advanced features. This manual is NOT a substitute for the documentation supplied by Apple. If you do not feel comfortable with the Apple ][, you may leave the Enhancer ][ installed but you should review your Apple ][ manuals before proceeding with this manual.

There is a glossary in the back of this manual. We have attempted to make it as comprehensive as possible. If you come across a word or term you aren't familiar with, refer to the glossary.

### Section 4.b Limitations of the Apple ][ Keyboard

The standard Apple ][ keyboard is an upper case only keyboard. Even though it is equipped with shift keys, it is incapable of lower case character entry. The shift keys are only used to shift the number and a few other keys. The Enhancer ][ changes all this so that the shift keys, and most other keys, may be put to greater use.

The standard Apple ][ keyboard is capable of entering only 91 of the possible 128 ASCII characters. The characters which cannot be entered are:

|    |      |                       |       |                  |
|----|------|-----------------------|-------|------------------|
| ~\ | <Fs> | (control backslash)   | {     | (left brace)     |
| ^_ | <Us> | (control underscore)  |       | (vertical bar)   |
| [  |      | (left square bracket) | }     | (right brace)    |
| \  |      | (backslash)           | ~     | (tilde)          |
|    |      | (underscore)          | <Rub> | (Rub-out)        |
| `  |      | (grave accent)        | a-z   | (all lower case) |

letters)

The Apple ][ keyboard does not have auto repeat nor fast repeat, as found on many computer terminals and some typewriters. Auto repeat means that when a key has been pressed and held down, after a brief pause it will repeat that character automatically.

The Apple ][ keyboard has a one character buffer. This means that you may type only one character while the computer is doing something else. If you type more than one character, the last character typed will be the character buffered (remembered).

The Apple ][ keyboard does not allow macro definitions. A keyboard with macro definitions would allow all or some keys or keystrokes to be defined as any combination of characters, up to a given limit.

#### Section 4.c What the Enhancer ][ Can Do

The Enhancer ][ completely changes your old Apple ][ keyboard into an intelligent keyboard. By installing the Enhancer ][, you have the potential to completely redefine the operation of your keyboard. The Apple ][ keyboard is divided into two parts: the keyboard switches and the encoder board (i.e. the electronics). The Enhancer ][ replaces the encoder board, so your keyboard is keeping its switches, but is getting a whole new brain. A much larger brain.

With the Enhancer ][, your shift keys become fully functional - like those of a typewriter [section: 5.e]. You may enter upper and lower case characters. You may enter any ASCII character, including those that the normal Apple ][ keyboard is incapable of entering [section: 4.b].

The Enhancer ][ gives your keyboard auto repeat and fast repeat [section: 5.a.3]. When you press and hold a key down for just less than a second, that key will begin to repeat at a rate of about 15 characters per second. If you press and hold the repeat (REPT) key down along with some other key simultaneously, the other key will be repeated at a faster rate, approximately 50 characters per second.

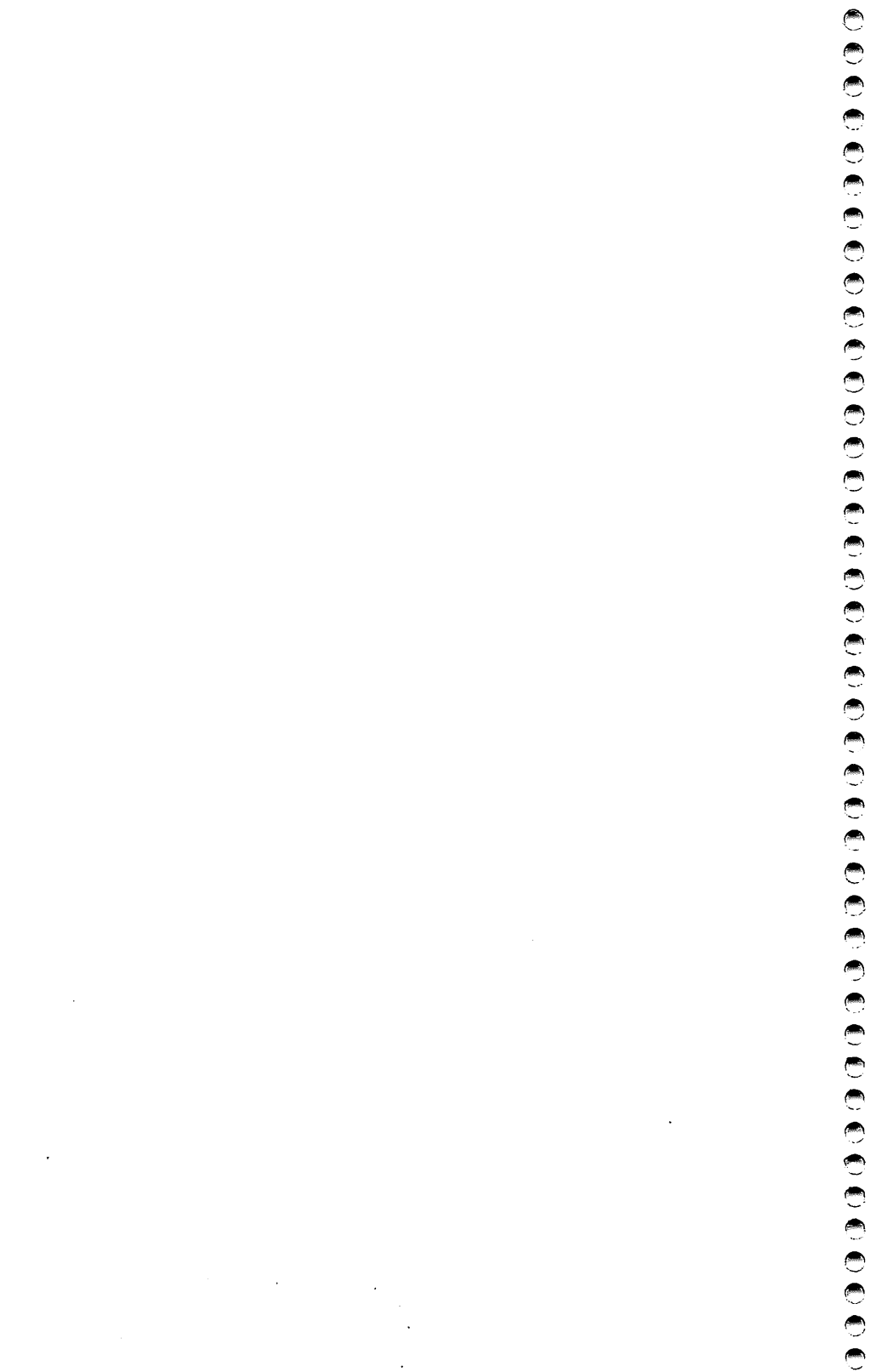
Since the Enhancer ][ has its own RAM, it is capable of remembering characters that you type while your Apple ][ is ignoring you (like when it's talking to the disc system). This is called a type ahead buffer [section: 5.g]. Naturally, the type ahead buffer is not unlimited. It has room for 128 characters. This is probably more than enough for most purposes.

## Chapter Four: A Primer for Beginners

The Enhancer ][ is also blessed with user definable keys - or macros [section: 5.f]. This means that you can equate a particular key with a character or sequence of characters up to 510 characters in length.

Your Enhancer ][ should have come with a chip labeled: "Lower Case". This chip may already be installed on your Apple ][. It cannot readily be seen without disassembly of your Apple ][. The function of this chip is to allow the display of lower case letters on your screen in the normal 40 column display format. If you have the Videoterm 80 column card, you do not need the Lower Case Chip to display lower case letters on the 80 column screen since the Videoterm has its own character generator.

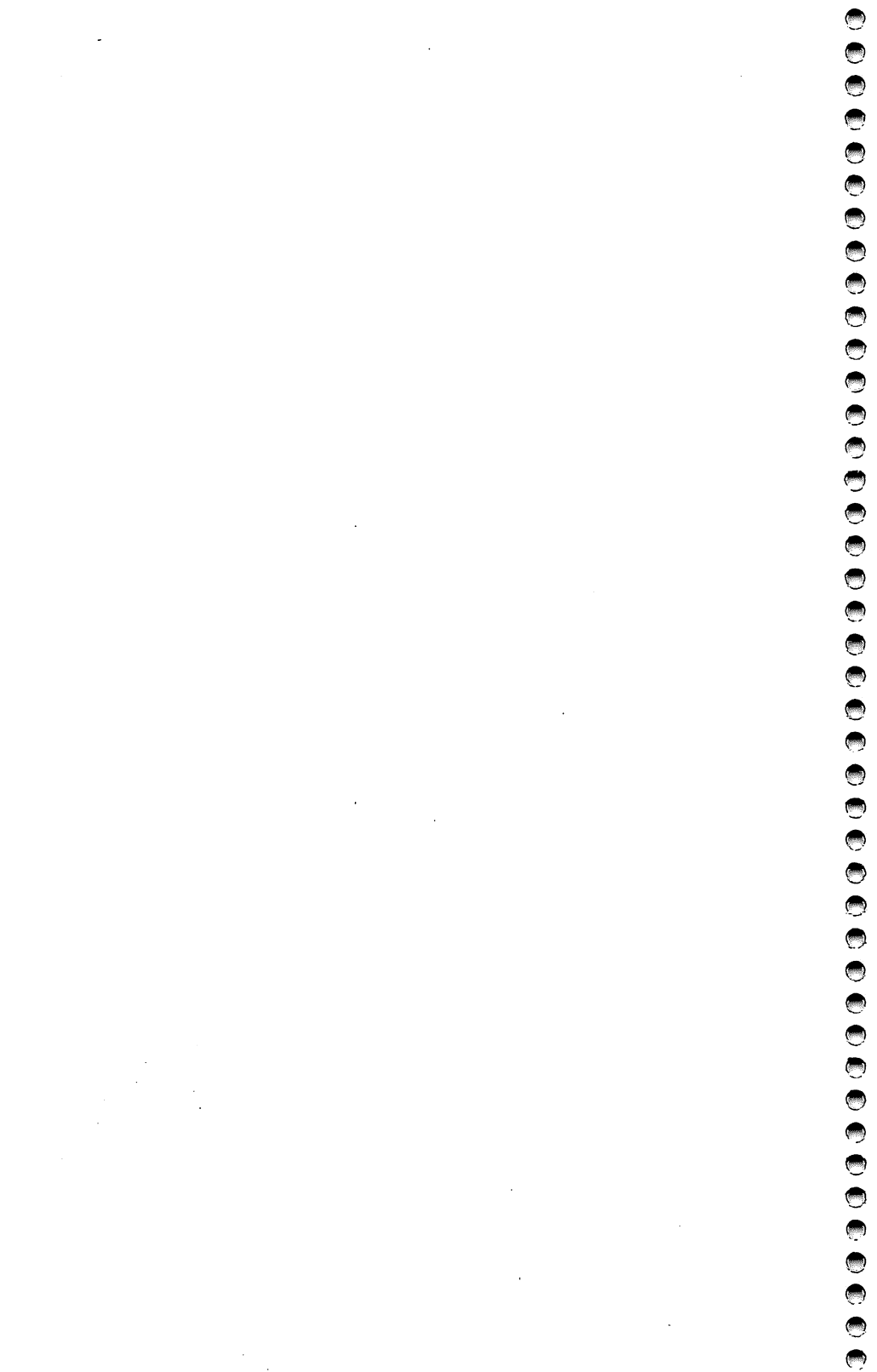
Note: Do NOT attempt to install the Lower Case Chip in a Revision 0 through 6 Apple. Attempting to do so can cause damage to your computer.



## Chapter Five: Operation

### Chapter Table of Contents.

- 5.a) Overview
  - 5.a.1) The Two Modes
  - 5.a.2) Macro Keys
  - 5.a.3) Auto & Fast Repeat
  - 5.a.4) The Type Ahead Buffer
- 5.b) Semantics
  - 5.b.1) ASCII Characters
  - 5.b.2) Keystrokes
  - 5.b.3) Function Keys and Character Keys
  - 5.b.4) Keyboard Characters
- 5.c) The Reset Key
  - 5.c.1) The System Reset
- 5.d) The Caps Lock Mode
- 5.e) The Lower Case (Caps Unlock) Mode
- 5.f) User Definable Macro Keys
  - 5.f.1) The Keys Which May Be Redefined
  - 5.f.2) Macro Memory Usage
  - 5.f.3) Defining a Macro
    - 5.f.3.1) Macro Definitions From the Keyboard
  - 5.f.4) Down Loading of User Defined Keys
  - 5.f.5) Repeat Reset
- 5.g) The Type Ahead Buffer
  - 5.g.1) The Acknowledge Line
  - 5.g.2) Macros
- 5.h) Self Test Diagnostics
- 5.i) Dvorak Option





## Chapter Five: Operation

### Section 5.a Overview

This chapter will discuss the operation of the Enhancer ][. This section is a brief overview of the chapter. Most of the questions that are raised as you read this section will probably be answered in subsequent sections. The Enhancer ][ is a sophisticated product and it will take some time for the user to become entirely familiar with its operation. Please do not feel intimidated if you find it necessary to re-read this chapter several times before all of the features are completely understood.

The Enhancer ][ has two modes of operation: The Caps Lock Mode, and the Lower Case Mode. If you think of your Apple ][ as having two separate keyboards, each functioning differently, you will begin to comprehend how the Enhancer ][ behaves. Each of these keyboards corresponds to one of the two modes.

#### Section 5.a.1 The Two Modes

The Caps Lock Mode behaves very much like the Apple's normal keyboard. Added are user definable macro keys, a type ahead buffer and auto repeat. In the Lower Case Mode, the shift key becomes fully functional for upper & lower case input and a Shift Lock feature is added. All 128 ASCII characters may be entered from this mode [section: 5.e].

As we begin to use advanced features, the two modes begin to diverge even farther. A macro defined in one mode will not be present in the other mode. It is as if your Apple had two physically different keyboards. This is a very important concept of this chapter.

Programmer's Note: A Mode Lock feature is selectable from down load [section: 5.f.4].

#### Section 5.a.2 Macro Keys

Later sections of this chapter will discuss exactly which key combinations may be defined as macros [section: 5.f.1]. For now, we will state that there are 376 unique keystroke combinations which may be defined. Exactly half of these keystrokes are found in each mode. If we define a keystroke combination to be equal to some macro in one mode, change modes and type the same keystroke combination, that macro will not exist in the second mode. Going back to our dual keyboard analogy, if we define some key on one keyboard, we would not expect it to have an effect upon the second keyboard. This is perhaps the single most important factor in understanding the use of macros.

## Chapter Five: Operation

### Section 5.a.3 Auto & Fast Repeat

Your keyboard is now capable of auto repeat. That is, when you press and hold a key down, after a brief pause (approximately 3/4 second), the character or macro associated with that key will begin to repeat. If you hold the repeat key down simultaneously, it will repeat at a faster rate. These two features are known as Auto Repeat and Fast Repeat respectively.

**Programmer's Note:** Auto and fast repeats may be disabled from a down load [section: 5.f.4].

### Section 5.a.4 The Type Ahead Buffer

The Enhancer ][ is equipped with a 128 character type ahead buffer. This means that you may type up to 128 characters while the Apple ][ is not scanning the keyboard (like when it's doing disc operations). When the Apple is ready for more keyboard input, the Enhancer ][ will tell the Apple exactly what you typed. Naturally, there will probably be times when you want to clear the buffer, so there are two flush buffer commands: ^C and Reset. More about this later [section: 5.g.2].

**Programmer's Note:** The type ahead buffer may be disabled from a down load [section: 5.f.4].

## Section 5.b Semantics

This section deals with some important definitions that will be used throughout this chapter. The reader is encouraged to understand the differences between the definitions of ASCII CHARACTER, KEYBOARD CHARACTER, KEYSTROKE, FUNCTION KEY and CHARACTER KEY. Once these definitions are clear, we will be better suited to understand the sometimes complex operation of macro key definitions and other functions.

### Section 5.b.1 ASCII Characters

The ASCII character set consists of 128 characters. Table 5.1 shows these characters and their corresponding decimal and hexadecimal values. With the Enhancer ][, you can enter any and all of these ASCII characters directly from your Apple ][ keyboard.

**Definition:** An ASCII CHARACTER is defined as a single element of Table 5.1.

## Chapter Five: Operation

Table 5.1 The ASCII Character Codes

| Decimal: |     | 128    | 144    | 160  | 176  | 192  | 208  | 224  | 240  |
|----------|-----|--------|--------|------|------|------|------|------|------|
| Hex:     |     | \$80   | \$90   | \$A0 | \$B0 | \$C0 | \$D0 | \$E0 | \$F0 |
| 0        | \$0 | ~@ Nul | ~P Dle |      | 0    | @    | P    | `    | p    |
| 1        | \$1 | ~A Soh | ~Q Dcl | !    | 1    | A    | Q    | a    | q    |
| 2        | \$2 | ~B Stx | ~R Dc2 | "    | 2    | B    | R    | b    | r    |
| 3        | \$3 | ~C Etx | ~S Dc3 | #    | 3    | C    | S    | c    | s    |
| 4        | \$4 | ~D Eot | ~T Dc4 | \$   | 4    | D    | T    | d    | t    |
| 5        | \$5 | ~E Enq | ~U Nak | %    | 5    | E    | U    | e    | u    |
| 6        | \$6 | ~F Ack | ~V Syn | &    | 6    | F    | V    | f    | v    |
| 7        | \$7 | ~G Bel | ~W Etb | '    | 7    | G    | W    | g    | w    |
| 8        | \$8 | ~H Bs  | ~X Can | (    | 8    | H    | X    | h    | x    |
| 9        | \$9 | ~I Ht  | ~Y Em  | )    | 9    | I    | Y    | i    | y    |
| 10       | \$A | ~J Lf  | ~Z Sub | *    | :    | J    | Z    | j    | z    |
| 11       | \$B | ~K Vt  | ^[ Esc | +    | ;    | K    | [    | k    | {    |
| 12       | \$C | ~L Ff  | ~\ Fs  | ,    | <    | L    | \    | l    |      |
| 13       | \$D | ~M Cr  | ~] Gs  | -    | =    | M    | ]    | m    | }    |
| 14       | \$E | ~N So  | ~^ Rs  | .    | >    | N    | ^    | n    | ~    |
| 15       | \$F | ~O Si  | ~_ Us  | /    | ?    | O    | _    | o    | rub  |

Note: HOW TO READ TABLE 5.1. The ASCII value of any character in table 5.1 may be determined by adding the value at the top of the column with the value to the left of the row that the character appears in. [A dollar sign (\$) preceding any value indicates hexadecimal. Values without a dollar sign are represented in decimal or base ten.] The first two columns of characters are the control characters. They are preceded by circumflexes (^) and followed by their ASCII names.

Example: A control G is represented by: ~G Bel. "Bel" is a short hand notation for "bell", meaning the bell character. Its ASCII value is \$87 (hexadecimal) or 135 (decimal).

Programmer Note: Table 5.1 shows the ASCII values with the high bit on. For equivalent ASCII representation with the high bit off, subtract \$80 (hexadecimal) or 128 (decimal).

### Section 5.b.2 Keystrokes

Definition: A KEYSTROKE is, for the purposes of this text, one and only one depression of any given key. Depressing two keys simultaneously constitutes two keystrokes.

Each character in Table 5.1 has a unique name and a corresponding ASCII value. Some ASCII characters may be entered

## Chapter Five: Operation

from the keyboard with only one keystroke (e.g. Return and Escape) while others may require as many as three keystrokes (e.g. the null character is a control-shift P). Regardless of how many keystrokes are required to type a particular ASCII character, it is considered to be only a single ASCII character. An ASCII character, therefore, is a single element of Table 5.1.

### Section 5.b.3 Function Keys and Character Keys

**Definition:** A FUNCTION KEY is a key that, when depressed by itself or in combination with another key(s), causes some function to occur. By themselves, function keys do not generate ASCII characters. The function keys are: the repeat, reset, control and shift keys. [Exception: depressing the Repeat and Reset keys simultaneously will cause the rub-out character to be generated.]

**Note:** Repeat - Reset should not be used to output a rub-out since this particular key sequence has other ramifications.

**Definition:** A CHARACTER KEY is any key on the Apple ][ keyboard which is NOT a function key.

The Apple ][ keyboard contains 52 keys, five of which are function keys (one repeat key, one reset key, one control key and two shift keys). The remaining 47 are character keys.

### Section 5.b.4 Keyboard Characters

**Definition:** A KEYBOARD CHARACTER is one, two or three keystrokes (of a mode) produced by a combination of one character key and one of the following four possible function key combinations: control, shift, control - shift or none. There are 376 unique keyboard characters, 188 in each mode.

**Example:** Depressing the shift and escape keys simultaneously produces an escape character. Depressing the escape key produces the escape character. Shift - Escape and Escape are two distinct keyboard characters. Normally, both of these keyboard characters produce the same ASCII character: <escape>.

**Another Example:** Depressing the Control and G keys simultaneously in the Caps Lock Mode produces a ^G. Depressing the Control and G keys simultaneously in the Lower Case Mode produces a ^G. A control G in Caps Lock Mode is a different keyboard character than a control G in the Lower Case Mode, however, both of these keystrokes produce the same ASCII character, namely a ^G.

## Chapter Five: Operation

The Apple ][ keyboard is capable of producing 188 unique keyboard character combinations in each mode. This number is calculated by the following formula:

$(47 \text{ character keys}) * 4 \text{ possible control-shift combinations}$

Where the 4 control-shift combinations are:

- i) A key may be pressed by itself
- ii) A key may be shifted (with either or both shift keys)
- iii) A key may be pressed in conjunction with the control key
- iv) A key may be control-shifted

There are a total of 376 keyboard characters. 188 in the Caps Lock Mode, 188 in the Lower Case Mode.

Table 5.2 [section: 5.d] lists the 47 character keys by their labels and the ASCII character returned by depressing these keys with or without the control or shift keys in the Caps Lock Mode. Table 5.3 [section: 5.e] is a similar table to Table 5.2 except that the characters listed are those of the Lower Case Mode. Table 5.4 [section: 5.e] depicts the differences between Table 5.2 & Table 5.3.

### Section 5.c The Reset Key

The Enhancer ][ uses the reset key as a command key. If, for example, the reset key is depressed by itself, the Enhancer ][ will go into Caps Lock Mode. If the shift and reset keys are depressed simultaneously, the Enhancer ][ will go into the Lower Case Mode.

Note: The above does not apply if the Mode Lock options has been selected during down load [section: 5.f.4].

#### Section 5.c.1 The System Reset

The Enhancer ][ has the ability to select one of the following system reset options:

- i) Control-Reset
- ii) Reset Only
- iii) Disable Reset

In the first setting, whenever the control and reset keys are depressed simultaneously, a system reset (processor reset) will occur. This mode is the default mode for which your Enhancer ][ has been configured at the factory. Most people prefer this configuration since it is less prone to accidental resets. In the second setting, a system reset occurs whenever

## Chapter Five: Operation

the reset key is depressed. If the Reset Only option has been selected, changing modes of the Enhancer ][ will also cause a system reset - generally not very useful. The third setting completely disables the system reset. With this option, it is still possible to change between the two modes.

Photo 2.11 shows the Enhancer ][. At the top right of the Enhancer ][, you will see a 6 pin Molex connector. The left three pins (1-3) are the reset control pins. When you examine your Enhancer ][, you should be able to see the plastic (probably blue) shorting blocks on pins 2 and 3. If you place this shorting block on pins 1 and 2, the Reset Only option is selected. Removing the block altogether totally disables the system reset.

### Section 5.d The Caps Lock Mode

The Enhancer ][ has two fundamental modes of operation. One is the Caps Lock Mode, the other is the Lower Case (caps unlock) Mode [section: 5.e]. With only one exception [section: 5.f.3.1], lower case letters may NOT be entered from Caps Lock Mode.

In the Caps Lock Mode, any keystroke sequence will produce the same character as would be produced on the standard Apple. The difference between the standard Apple and the Enhancer ][ in Caps Lock Mode is the auto and fast repeats, the user definable macros and the type ahead buffer. Table 5.1 shows all the possible combinations of keys and the ASCII values generated by typing them.

## Chapter Five: Operation

Table 5.2 Keyboard Characters and their Associated ASCII Characters In the Caps Lock Mode

| Key        | Alone | Cntl | Shift | Both | Key         | Alone | Cntl | Shift | Both |
|------------|-------|------|-------|------|-------------|-------|------|-------|------|
| Space      |       |      |       |      | Return      | ~M    | ~M   | ~M    | ~M   |
| 0          | 0     | 0    | 0     | 0    | G           | G     | ~G   | G     | ~G   |
| 1          | 1     | 1    | !     | !    | H           | H     | ~H   | H     | ~H   |
| 2          | 2     | 2    | "     | "    | I           | I     | ~I   | I     | ~I   |
| 3          | 3     | 3    | #     | #    | J           | J     | ~J   | J     | ~J   |
| 4          | 4     | 4    | \$    | \$   | K           | K     | ~K   | K     | ~K   |
| 5          | 5     | 5    | %     | %    | L           | L     | ~L   | L     | ~L   |
| 6          | 6     | 6    | &     | &    | M           | M     | ~M   | ]     | ~]   |
| 7          | 7     | 7    | '     | '    | N           | N     | ~N   | ^     | ~^   |
| 8          | 8     | 8    | (     | (    | O           | O     | ~O   | o     | ~o   |
| 9          | 9     | 9    | )     | )    | P           | P     | ~P   | @     | ~@   |
| :*         | :     | :    | *     | *    | Q           | Q     | ~Q   | q     | ~q   |
| ; <b>+</b> | ;     | ;    | +     | +    | R           | R     | ~R   | r     | ~r   |
| ,<         | ,     | ,    | <     | <    | S           | S     | ~S   | s     | ~s   |
| =          | =     | =    | =     | =    | T           | T     | ~T   | t     | ~t   |
| .>         | .     | .    | >     | >    | U           | U     | ~U   | u     | ~u   |
| /?         | /     | /    | ?     | ?    | V           | V     | ~V   | v     | ~v   |
| A          | A     | ~A   | A     | ~A   | W           | W     | ~W   | w     | ~w   |
| B          | B     | ~B   | B     | ~B   | X           | X     | ~X   | x     | ~x   |
| C          | C     | ~C   | C     | ~C   | Y           | Y     | ~Y   | y     | ~y   |
| D          | D     | ~D   | D     | ~D   | Z           | Z     | ~Z   | z     | ~z   |
| E          | E     | ~E   | E     | ~E   | Right Arrow | ~U    | ~U   | ~U    | ~U   |
| F          | F     | ~F   | F     | ~F   | Left Arrow  | ~H    | ~H   | ~H    | ~H   |
|            |       |      |       |      | Escape      | ~[    | ~[   | ~[    | ~[   |

### Section 5.e The Lower Case (Caps Unlock) Mode

The Enhancer, ][ has two fundamental modes of operation. One is the Lower Case (caps unlock) Mode, the other is the Caps Lock Mode [section: 5.d]. With only one exception [section: 5.f.3.1], lower case letters may ONLY be entered from the Lower Case Mode.

In the Lower Case Mode, all 128 ASCII characters may be entered directly from the keyboard. Naturally, user definable macros, a type ahead buffer and auto and fast repeats are available. Table 5.3 shows all the possible combinations of keys and the ASCII values generated by typing them.

## Chapter Five: Operation

### Section 5.e.1 Shift Lock

When in the Lower Case Mode, if you depress the control key BY ITSELF, you will be placed in shift lock mode. In this mode, if you depress any key, the character output will be the same as if the shift key were held down. This mode will be maintained until either shift key is pressed.

Table 5.3 Keyboard Characters and their Associated ASCII Characters in the Lower Case Mode

| Key   | Alone | Cntl | Shift | Both | Key         | Alone | Cntl | Shift | Both |
|-------|-------|------|-------|------|-------------|-------|------|-------|------|
| Space |       |      |       |      | Return      | ~M    | ~M   | ~M    | ~M   |
| 0     | 0     | ~@   | @     | ~@   | G           | g     | ~G   | G     | ~G   |
| 1     | 1     | ~    | !     | ~    | H           | h     | ~H   | H     | ~H   |
| 2     | 2     | ~"   | "     | ~"   | I           | i     | ~I   | I     | ~I   |
| 3     | 3     | rub  | #     | rub  | J           | j     | ~J   | J     | ~J   |
| 4     | 4     | ~\   | \$    | ~\   | K           | k     | ~K   | K     | ~K   |
| 5     | 5     | ~]   | %     | ~]   | L           | l     | ~L   | L     | ~L   |
| 6     | 6     | ~^   | &     | ~^   | M           | m     | ~M   | M     | ~M   |
| 7     | 7     | ~'   | '     | ~'   | N           | n     | ~N   | N     | ~N   |
| 8     | 8     | ~{   | (     | ~{   | O           | o     | ~O   | O     | ~O   |
| 9     | 9     | ~}   | )     | ~}   | P           | p     | ~P   | P     | ~P   |
| :*    | :     | ~*   | *     | ~*   | Q           | q     | ~Q   | Q     | ~Q   |
| ;;+   | ;     | ~+   | +     | ~+   | R           | r     | ~R   | R     | ~R   |
| ,<    | ,     | ~<   | <     | ~<   | S           | s     | ~S   | S     | ~S   |
| -_=   | -     | ~_   | =     | ~_   | T           | t     | ~T   | T     | ~T   |
| .>    | .     | ~>   | >     | ~>   | U           | u     | ~U   | U     | ~U   |
| /?    | /     | ~?   | ?     | ~?   | V           | v     | ~V   | V     | ~V   |
| A     | a     | ~A   | A     | ~A   | W           | w     | ~W   | W     | ~W   |
| B     | b     | ~B   | B     | ~B   | X           | x     | ~X   | X     | ~X   |
| C     | c     | ~C   | C     | ~C   | Y           | y     | ~Y   | Y     | ~Y   |
| D     | d     | ~D   | D     | ~D   | Z           | z     | ~Z   | Z     | ~Z   |
| E     | e     | ~E   | E     | ~E   | Right Arrow | ~U    | ~U   | ~U    | ~U   |
| F     | f     | ~F   | F     | ~F   | Left Arrow  | ~H    | ~H   | ~H    | ~H   |
|       |       |      |       |      | Escape      | ~[    | ~[   | ~[    | ~[   |

**Example:** Depress the control key BY ITSELF. After releasing the control key, depress the K key. An upper case K will appear. Now depress the comma key, a less than sign will appear on the screen. Numbers will also be shifted. Depressing the shift key (and releasing) and depressing the K key again now yields a lower case K.

**Note:** Shift Lock does NOT apply to macros either during definition nor use.



## Chapter Five: Operation

Example: Suppose that the K key had been redefined as some other character. Now if we depressed the control key by itself, we would be in shift lock mode. If we depressed any key except the K key, it would be shifted. The K key, because it has a macro associated with it, is not effected by the Shift Lock Mode. This is true even if the K key were redefined as a K.

Table 5.4 Differences Between Table 5.2 and Table 5.3 using the Values of Table 5.3

| Key   | Alone | Cntl | Shift | Both | Key         | Alone | Cntl | Shift | Both |
|-------|-------|------|-------|------|-------------|-------|------|-------|------|
| Space |       |      |       |      | Return      |       |      |       |      |
| 0     |       | ~@   | @     | ~@   | G           | g     |      |       |      |
| 1     |       |      |       |      | H           | h     |      |       |      |
| 2     |       | ~    |       | ~    | I           | i     |      |       |      |
| 3     |       | rub  |       | rub  | J           | j     |      |       |      |
| 4     |       | ~\   |       | ~\   | K           | k     |      |       |      |
| 5     |       | ~]   |       | ~]   | L           | l     |      |       |      |
| 6     |       | ~~   |       | ~~   | M           | m     |      | M     | ~M   |
| 7     |       | ,    |       | ,    | N           | n     |      | N     | ~N   |
| 8     |       | {    |       | {    | O           | o     |      |       |      |
| 9     |       | }    |       | }    | P           | p     |      | P     | ~P   |
| :*    |       | ~    |       | ~    | Q           | q     |      |       |      |
| ++;   |       | ~    |       | ~    | R           | r     |      |       |      |
| ,<    |       | [    |       | [    | S           | s     |      |       |      |
| =     |       | ~    |       | ~    | T           | t     |      |       |      |
| .>    |       | ]    |       | ]    | U           | u     |      |       |      |
| /?    |       | \    |       | \    | V           | v     |      |       |      |
| A     | a     |      |       |      | W           | w     |      |       |      |
| B     | b     |      |       |      | X           | x     |      |       |      |
| C     | c     |      |       |      | Y           | y     |      |       |      |
| D     | d     |      |       |      | Z           | z     |      |       |      |
| E     | e     |      |       |      | Right Arrow |       |      |       |      |
| F     | f     |      |       |      | Left Arrow  |       |      |       |      |
|       |       |      |       |      | Escape      |       |      |       |      |

### Section 5.f User Definable Macro Keys

With your Enhancer ][ you may redefine up to 170 single character macros or one 510 character macro. A macro is a keyboard character which has been redefined to be any ASCII character, or sequence of ASCII characters up to 510 ASCII characters in length. Careful reading of this section and section 5.b is recommended if you plan to use this feature as it is rather complex.

## Chapter Five: Operation

You have already learned how to issue some of the commands that you may give the Enhancer ][ by using the reset and shift keys. In this section you will learn how to use new commands utilizing the repeat key.

### Section 5.f.1 The Keys Which May Be Redefined

The Enhancer ][ has two modes of operation [section: 5.d & 5.e]. Each keyboard character in one mode is distinct from any keyboard character in the other mode [section: 5.b.4]. Since there are 188 keyboard characters possible from the Apple ]['s keyboard, there are exactly 376 unique keyboard characters possible in both the modes of operation [section: 5.b.4]. Any of these 376 keyboard characters may be redefined. A definition may be from one to 510 ASCII characters in length.

Example: Suppose the Enhancer ][ is in the Caps Lock Mode and that we have redefined the shift C keyboard character to be: CATALOG. Now if one depressed the shift C, CATALOG would be printed on the screen. If we entered the Lower Case Mode and depressed shift C, a shifted C would appear on the screen. Returning to the Caps Lock Mode, and depressing a shift C would, once again, cause CATALOG to be printed. Depressing the C key by itself will produce a capital C.

### Section 5.f.2 Macro Memory Usage

The Enhancer ][ has 512 bytes of memory reserved for macro definitions. Each macro requires two bytes overhead and one byte for each ASCII character of the macro. The following formula determines the number of macros possible given the average size of each macro:

$$\text{Number of Macros} = \frac{512}{2 + \text{Average Size of Macros}}$$

Likewise, the average size may be computed by:

$$\text{Average Size of Macros} = \frac{512}{\text{Number of Macros}} - 2$$

## Chapter Five: Operation

### Section 5.f.3 Defining a Macro

Macros may be defined in two ways:

- 1) From the Keyboard
- ii) Down Loaded from a Disc File

Note: Both methods of Macro definitions may be selectively disabled from a down load [section: 5.f.4].

#### Section 5.f.3.1 Macro Definitions From the Keyboard

Probably the best way to describe how a macro may be defined from the keyboard is to start with an example. Let us redefine a shift C keyboard character to be: CATALOG <cr>. Here are the steps we would preform:

| Keystroke Sequence           | Characters Output from the Keyboard |
|------------------------------|-------------------------------------|
| Step 1: Control-shift-repeat | <None>                              |
| Step 2: Shift C              | <None>                              |
| Step 3: CATALOG <cr>         | CATALOG <cr>                        |
| Step 4: Repeat               | <None>                              |

Step 1 tells the Enhancer ][ that we want to define a macro. At step 2, we enter the keyboard character that we wish to define. Step 3 is to enter the macro itself and at Step 4 we tell the Enhancer ][ that we wish to end the macro definition by depressing the repeat key. Notice that the only step which sends any ASCII character(s) to the Apple ][ is Step 3. The characters of the macro definition are the only characters which are output from the keyboard during a macro definition. [Please note that these characters may or may not be output to your monitor, depending upon what software is active at the time.] Step 4 ends the macro definition.

If you depress the shift, control and repeat keys in conjunction with a character key, the corresponding ASCII character or macro definition is fast repeated, and a macro definition is NOT begun.

Note: When fast repeating a character which is control - shifted, it is recommended that the repeat key be the last key pressed and the first key released to eliminate any keyboard bounce.

## Chapter Five: Operation

Here is an example of how to clear a macro key definition or abort a definition once you have depressed the shift, control and repeat keys simultaneously:

| Keystroke Sequence           | Characters Output from the Keyboard |
|------------------------------|-------------------------------------|
| Step 1: Control-shift-repeat | <None>                              |
| Step 2: Space Bar            | <None>                              |
| Step 3: Repeat               | <None>                              |

Step 1 begins the macro definition. At Step 2 you enter the macro to be cleared. If you wish to abort the definition sequence once you have begun, you must enter some keyboard character. You might want to use some key that you have not already defined (perhaps the space bar). Step 3 is to end the macro definition without entering any macro. This procedure will clear any macro associated with the keyboard character in Step 2.

Warning: Omitting Step 2 will not abort a macro definition.

Note: You cannot use the repeat key when defining a macro. You may, however, use the auto repeat function by pressing and holding a key down.

When you are defining a macro, all other macros are temporarily disabled. Recursive macros are not allowed, that is, a macro cannot call another macro. A macro is defined as being some set of ASCII characters, not keyboard characters.

Mode changes are legal when defining a macro. It is possible, therefore, to define a macro in Caps Lock Mode using lower case characters. Changing modes within a macro does not take extra memory to store that macro since only the ASCII values are stored in the macro definition, not the mode actually used to generate the character. Likewise, use of a macro will NEVER cause the mode of your keyboard to change. If you begin a macro in one mode, change the mode during the definition and end the definition, your keyboard will be in the second mode once the definition is complete. Your macro, however, will only be defined in the first mode, the mode in which you are not (in this scenario).

Example: You are in the Caps Lock Mode. You depress the shift, control and repeat keys simultaneously. Now you depress the shift and L keys simultaneously. You have just begun to define a shift L in Caps Lock Mode. You now depress the shift

## Chapter Five: Operation

and reset keys simultaneously. This puts you into the Lower Case Mode. You type: "Lower Case Characters" <repeat>. The <repeat> means that you have depressed the repeat key. The definition is now complete. Depressing another character, you find that you are still in the Lower Case Mode. You depress the shift L and a capital L appears on the screen. You now return the Caps Lock Mode by depressing the reset key. Depressing shift L, you find "Lower Case Characters" displayed on your screen.

### Section 5.f.4 Down Loading of User Defined Keys.

It is possible to program your keyboard's macro definitions directly from a disc file without having to enter them from the keyboard. Two programs have been provided on the Enhancer ][ Utilities Disc for this purpose: the Macro Editor [section: X.d] and the Down Load Program [section: X.e]. You may create disc files containing different macro definitions for various purposes using the Macro Editor and down load them into your keyboard anytime by using the Down Load Program. You may therefore establish a turnkey system where your keyboard will automatically be programmed by your hello program each time DOS is booted.

Note: Macros CANNOT be down loaded from disc unless the acknowledge wire has been installed [section: 5.g.1].

There are two methods of beginning the down load process. The first is by depressing the repeat and reset keys simultaneously. The second is by installing a one wire modification from pin 12 of F14 on the motherboard to the keyboard connector socket. When this is done, down loading can be done without any operator intervention (i.e. automatically). If the Enhancer ][ receives the signal to begin down load but the down load program is not running, the down load is aborted, however all macro definitions are cleared.

Down loading has certain advantages over keyboard entry. You will probably become accustomed to special macros that you will want to have all the time. Down loading is a quick and easy way to enter them into your Enhancer ][. There are certain other options which normally are not available from the keyboard alone. These include the following:

- i) Disable Shift Lock
- ii) Lock Keyboard Mode
- iii) Select Mode After Down Load
- iv) Disable Auto Repeat
- v) Disable Type Ahead
- vi) Disable Keyboard Macro Entry & Clear
- vii) Disable Auto Down Load

## Chapter Five: Operation

For more information, refer to Appendix C and section X.d.

### Section 5.f.5 Repeat Reset

Depressing the repeat and reset keys simultaneously will clear all macro definitions and produce a rub-out character to be output from the keyboard. It will also begin the down load process IF the down load program is running.

Note: Use of the repeat - reset to produce a rub-out character is discouraged due to the other effects.

### Section 5.g The Type Ahead Buffer

The type ahead buffer is useful in many applications, especially when the microprocessor is busy with tasks other than checking the keyboard output, such as disc operations. To take full advantage of this feature, the acknowledge wire should be installed (between sockets A7 and B7 on the motherboard). The Enhancer ][ is programmed to give a pseudo type ahead buffer in the absence of this wire, however this operation is discouraged due to the possibility of data loss.

Programmer Note: The type ahead buffer may be disabled from a down load [section: 5.f.4].

#### Section 5.g.1 The Acknowledge Line

Normally, when the keyboard has a character to send, it sends it. When the Apple receives a character from the keyboard, it will set a strobe. This means that programs know when a character is waiting to be read from the keyboard. The keyboard, on the other hand, does not know when that character has been processed by the Apple. To make a type ahead buffer function efficiently, it must know when the character has been read so that it can send the next character. The acknowledge wire gives this information to the Enhancer ][. Without the acknowledge wire, the Enhancer ][ has no way of knowing when the Apple has read the data off the keyboard and is ready for more.

Note: The Enhancer ][ is programmed to send characters out at certain intervals IF the acknowledge wire has NOT been installed, in the hope that the Apple will be ready to receive the next character. Since many factors can determine when the Apple is ready for a character, this method is prone to data loss. The Enhancer ][ has been programmed using techniques which will minimize data loss without causing a severe delay in overall throughput.

## Chapter Five: Operation

Installation of the acknowledge wire [section: 2.d] is neither very difficult nor time consuming. The time taken to install it will pay for itself many times over and will significantly improve the reliability and efficiency of your Enhancer ][.

Note: If the acknowledge wire is not installed, macros CANNOT be down loaded from disc.

### Section 5.g.2 The Flush Buffer Commands

There are two commands which will flush (empty) the type ahead buffer. They are Reset and ^C. A flush buffer command is necessary for a number of reasons, the most significant of which is to clear the buffer once you no longer need its contents or when you believe that you may have entered garbage into it.

Control C was chosen as a flush buffer command because of BASIC. If ^C did not flush the buffer, BASIC wouldn't see the ^C if there was any character in the buffer before the ^C and you wouldn't be able to halt program execution without a system reset. Since it might be useful to enter a ^C into a macro, ^C will not flush the buffer IF the ^C is PART of the macro definition.

Note: If you redefine the Control-C keyboard character to be a ^C ASCII character, depressing Control-C will not cause the buffer to be flushed (in the mode defined) since it is a macro. If that macro is cleared, ^C will once again cause a flush buffer.

Note: Changing modes, or depression of the Reset key at any time will cause the buffer to be flushed.

### Section 5.g.3 Macros

Macros use the type ahead buffer. When a macro key is struck, that macro is placed into the type ahead buffer. If the buffer is filled, the Enhancer ][ will wait until a character is output from the buffer before placing the next character of the macro into the buffer.

Note: If the acknowledge wire has not been installed or the type ahead buffer has been disabled, macros MAY appear not to function properly due to data loss.

Note: If the acknowledge wire is not installed, macros CANNOT be down loaded from disc.

## Chapter Five: Operation

### Section 5.h Self Test Diagnostics

Whenever the power is turned on to your Apple ][ computer, your Enhancer ][ performs a self test. The entire test takes less than a second but is rather comprehensive. The RAM is tested and a checksum is performed on the firmware. If any fault conditions are found, the Enhancer ][ will attempt to print an error message to your screen. Naturally, there can be conditions where output of an error message is impossible.

To initiate the self test routine manually, turn your Apple ]['s power off and remove any disc controller card you may have in the system. Now turn the power on. If no error message is printed and you can enter characters into the system normally, your Enhancer ][ is probably in good working order. If an error message is printed, refer to chapter three.

Generally, if errors which are subtle in nature do occur in the Enhancer ][, they will be detected by the self test diagnostics. Gross errors will probably lock your keyboard up entirely. Therefore, if no error message is printed on power up, and your keyboard allows you to enter characters, it is very likely that it is functioning correctly. If you feel, however, that there is something wrong, refer to chapter three.

Note: The Enhancer ]['s RAM and ROM are entirely separate from any of your Apple ]['s memory (both RAM & ROM). You CANNOT make a memory listing of your Enhancer ][ from your Apple ][.

### Section 5.i Dvorak Option

As lore would have it, the Dvorak keyboard layout [photo: 5.i.1] was the result of a United States government grant - the purpose of which was to find the most efficient keyboard layout possible. Apparently, the beloved QWERTY keyboard, with which the Apple ][ is blessed, was designed to actually slow the typist down. This is because the first typewriters could not type as fast as the first typists could strike the keys, causing damage to their primitive mechanisms. Thus the QWERTY layout.

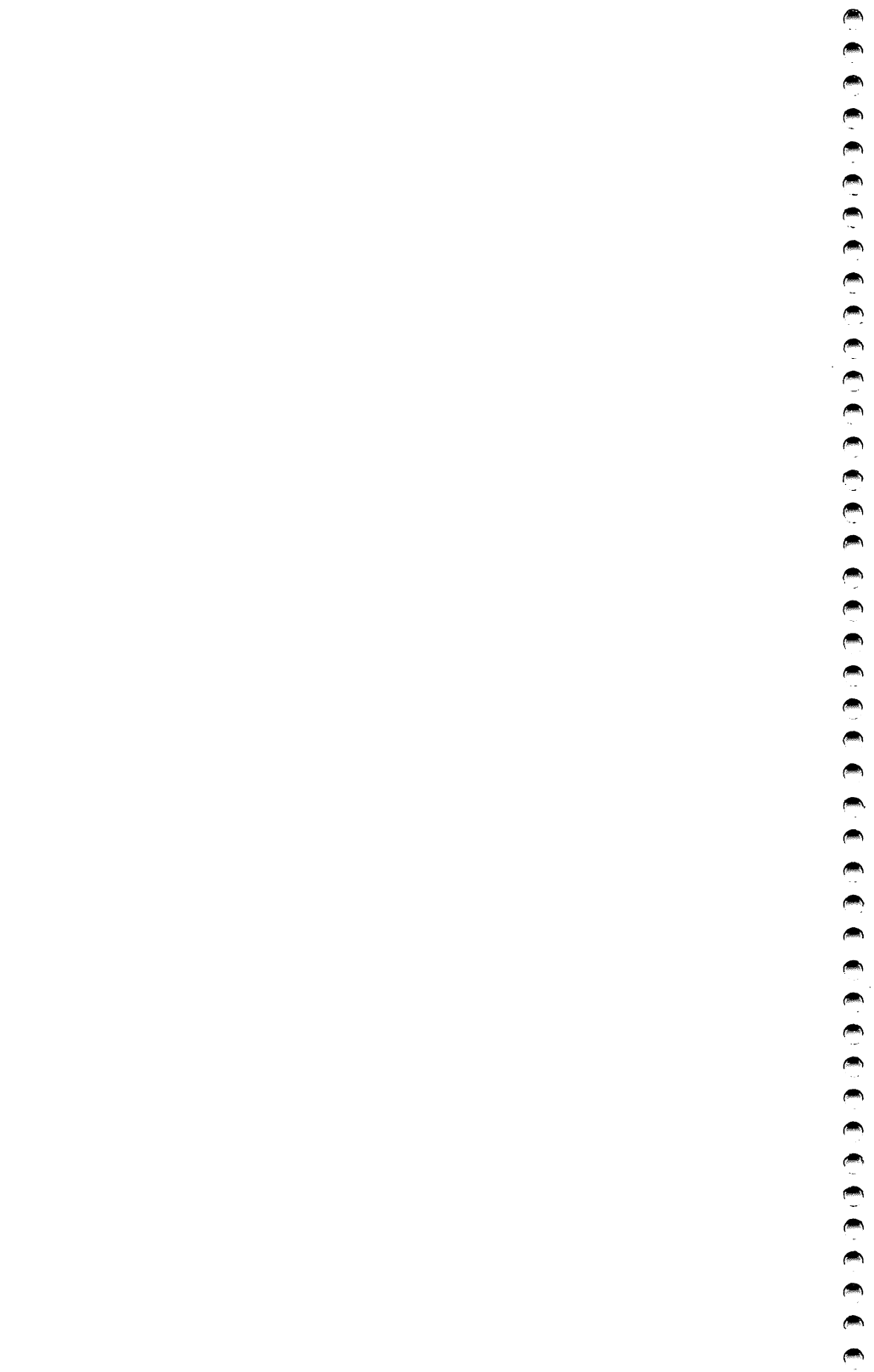
The Dvorak keyboard is a keyboard in which all the vowels are located on the home row of keys. Supposedly all the other keys are placed in such a way that the most frequently used keys are easiest to get to. Proponents say that the Dvorak keyboard can be learned 2 to 4 times more quickly than the QWERTY keyboard. Typing speed is supposed to be increased by about 70%.

Since June of 1981, Videx has supported a Dvorak option for the original Keyboard & Display Enhancer and will continue this support for the Enhancer ][. An optional firmware EPROM which will remap your keyboard into the Dvorak layout may be purchased



## Chapter Five: Operation

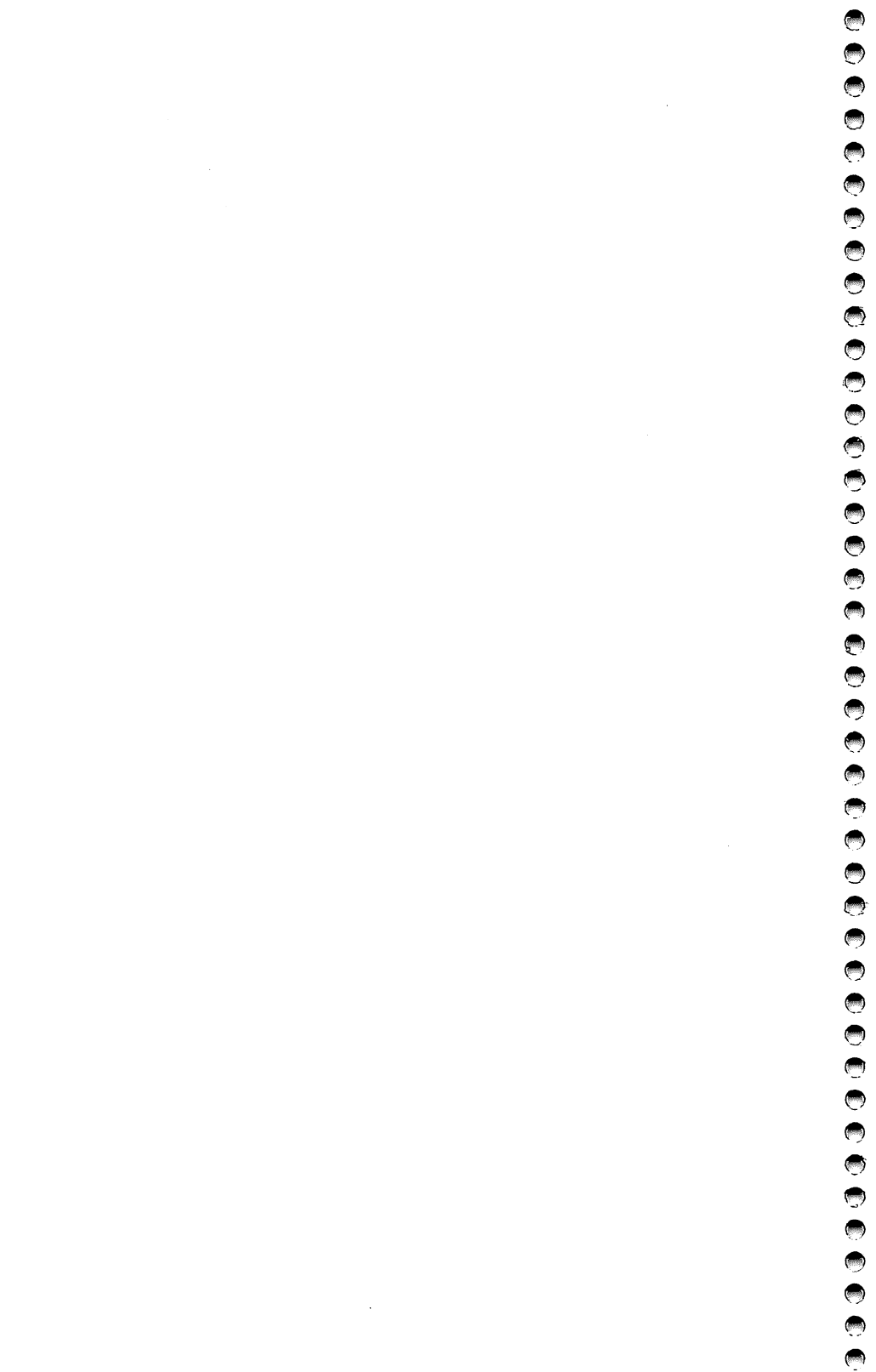
from Videx. The user may then pull the key-caps off their keyboard and change the layout of the keys themselves.



## Chapter Six: Apple ][ Language Considerations

### Chapter Table of Contents.

- 6.a) Apple DOS & BASICs
  - 6.a.1) The RAM Card Solution
  - 6.a.2) ROM Cards
  - 6.a.3) The Key Filter Program
- 6.b) 6502 Machine Language
- 6.c) Pascal
- 6.d) FORTRAN



## Chapter Six: Apple ][ Language Considerations

### Section 6.a Apple BASICs and DOS

In both of the Apple ]['s monitor ROMs (old and new), there is a routine that converts lower case characters to upper case characters. This routine, named CAPTST, also converts certain symbols into other characters [appendix: A]. Unfortunately, CAPTST is used by most of the major input routines of the monitor, DOS and both BASICs (including BASIC's INPUT statement). Most lower case characters entered into the computer will therefore be mapped into upper case characters unless CAPTST is either altered or bypassed. The Videoterm 80 column board, when active, will bypass CAPTST. The text of this section, therefore, applies only to the 40 column mode.

In this section, we will discuss software and hardware methods of bypassing or altering CAPTST. Not all methods may be used by all users. Please be careful to note any of the cautions listed for the method you choose to use.

The optimal solution for your system configuration depends upon whether or not it includes a RAM card, either Apple's Language Card or some other brand.

#### Section 6.a.1 The RAM Card Solution

There are two types of RAM cards: those with on-board F8 ROMs and those without. The Apple Language Card contains this ROM, most of the others do not.

If your card does have a socket for a 2716 EPROM to replace the F8 ROM, you are advised to take advantage of this feature. Here is what it does: the on-board F8 ROM completely replaces the F8 ROM on the motherboard. This means that the motherboard F8 ROM is permanently disabled. It may be removed from the system completely. If the RAM is active, both the on-board F8 ROM and the motherboard F8 ROM are disabled. If you do replace the on-board F8 ROM with your own copy, your resident language can contain the Lower Case Fix. If you do not replace this ROM, your system will behave just like the RAM cards without the on-board F8 ROMs.

RAM cards without on-board F8 ROMs will not be able to implement the Lower Case Fix in the resident BASIC. You may, however, load either BASIC into the RAM card and once there, make the Lower Case Fix. The Configuration Program on the Enhancer ][ Utilities Disc will create modified FPBASIC and INTBASIC files and set the HELLO program to load one of the two BASICs into your RAM card every time you boot to that disc. When you run the configuration program [section: X.a], you should select the version of BASIC that you use the most to be loaded into the RAM card.

## Chapter Six: Apple ][ Language Considerations

Note: If you have an Apple ][ plus system, and you configure your hello program to load Applesoft into the RAM card, Integer BASIC will not be immediately available to you. That is, if you try to RUN an Integer BASIC file, or type in INT, you will get a LANGUAGE NOT AVAILABLE error message. To use Integer BASIC, you need to type: BRUN INTBASIC. At this point, Integer BASIC is available, however Applesoft will no longer be capable of lower case input since your system will use the ROM version. To return to a Lower Case Fixed version of Applesoft, you should BRUN FPBASIC followed by an attempt to enter Integer BASIC (i.e. type INT). This has to do with the version of Applesoft at which DOS is currently looking.

### Section 6.a.2 ROM Cards

Owners of ROM cards may program and modify a 2716 EPROM to allow lower case input in both BASICs [section: A.c].

### Section 6.a.3 The Key Filter Program

There is a program on the Enhancer ][ Utilities Disc called Key Filter. Key Filter is a software method of bypassing CAPTST. It can be used to allow input of lower case characters by most programs written in either BASIC or 6502 machine language. The Key Filter program should not be used with word processors [section: 7.a].

Since the Key Filter program lives in RAM, it will take up some of your free memory - though not a great deal. It is vulnerable to attack from other programs. It can be disconnected with certain monitor, DOS and BASIC commands. It must be loaded from disc each time the power is turned on or DOS is booted. Key Filter is a good solution for many people, however, there are other solutions as well.

### Section 6.b 6502 Machine Language

When writing programs in 6502 machine code, if you do not use the Apple ][ monitor's GETLN routine (including GETLNZ & GETLNI), you should not encounter problems with lower case input. We suggest that you write your own GET LINE routine. If you should choose not to, CAPTST must be modified. The procedure is the same as that which is listed in the Apple DOS and BASICS section [section: 6.a].

The single character input routines of the monitor, namely KEYIN, RDKEY and RDCHAR have no problems inputting any ASCII character, including lower case characters.

Direct reading of the keyboard causes no problems.

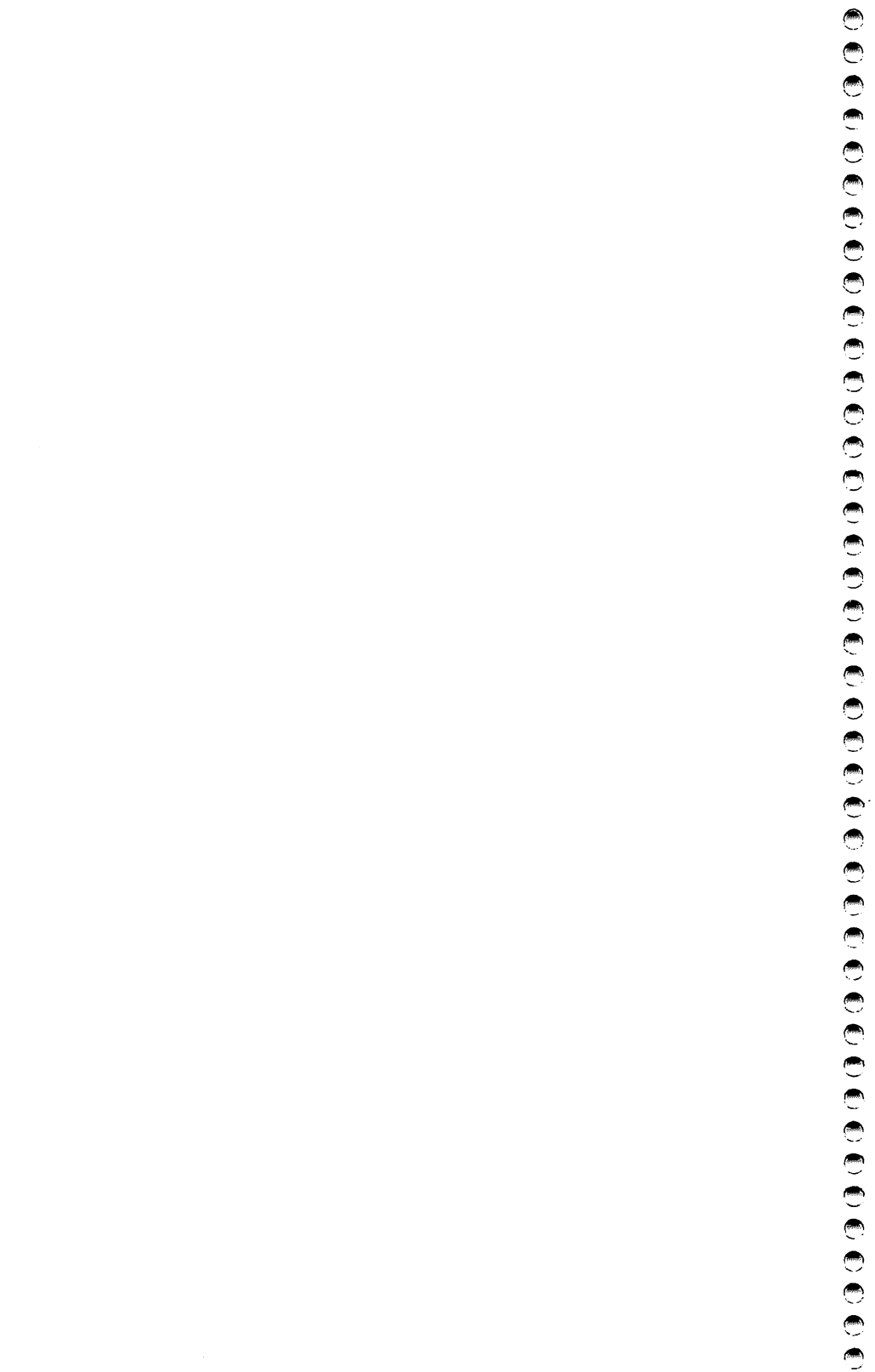
## Chapter Six: Apple ][ Language Considerations

### Section 6.c Pascal

Pascal has no problems with lower case letters if you are using the Videoterm 80 column card or once SYSTEM.APPLE has been modified by OUTPATCH (40 columns) [section: X.f].

### Section 6.d FORTRAN

Apple FORTRAN uses the Apple Pascal Operating System. Therefore, if you have a Videoterm or, in 40 columns, if you use a Outpatched [section: X.f] version of the SYSTEM.APPLE file (which you might be using for your Pascal discs) on your FORTRAN discs, Apple FORTRAN will have no more problems with lower case characters than Pascal does, which presumably is none. If you are having problems, refer to the Pascal section of this manual [section: 6.c].

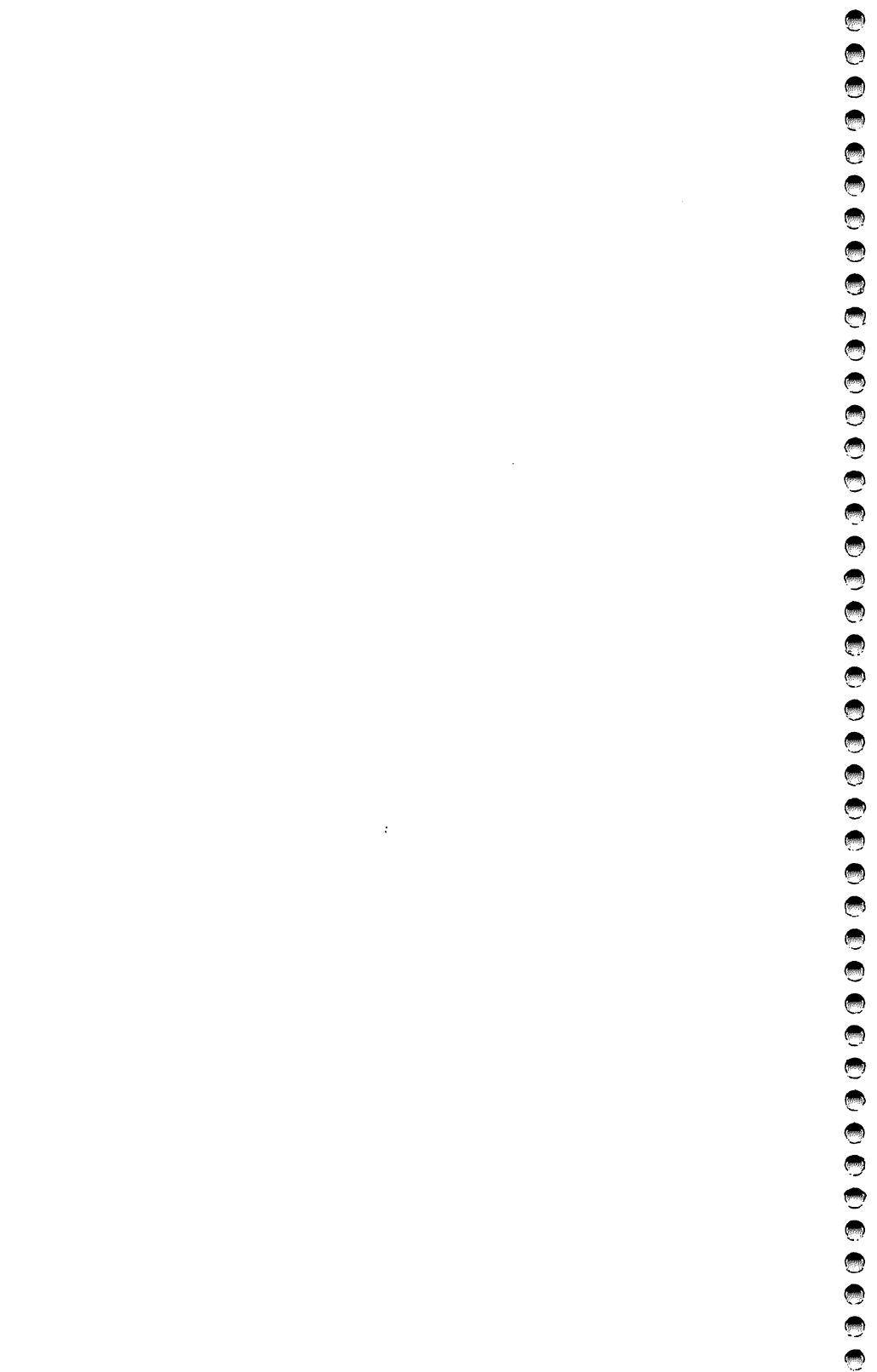




Chapter Seven: Other Software Considerations

Chapter Table of Contents.

- 7.a) Word Processors
  - 7.a.1) Generally
  - 7.b.2) A Note on Wordstar
- 7.b) CP/M
- 7.c) Other Commercial Software
  - 7.c.1) Applesoft, Integer BASIC and DOS
  - 7.c.2) Pascal etc.
  - 7.c.3) CP/M



## Chapter Seven: Other Software Considerations

### Section 7.a Word Processors

Most word processors will work with the Enhancer ][. Some have a special configuration for hardware input of lower case. Some have various other options available. This section presents some general guidelines to follow.

#### Section 7.a.1 Generally

Almost all word processors have some kind of configuration program or routine. You should read your word processing system's manual to determine the various options available to you. Since each word processing system is different, we will present some of the options which are usually available and whether or not they are appropriate.

The Enhancer ][ DOES have lower case (hardware) DISPLAY capability.

The Enhancer ][ DOES have lower case (hardware) INPUT capability.

The Enhancer ][ is NOT a shift wire, single wire or single wire shift mod. It does not use any game paddle port nor push button input. Answering yes to this option often inhibits input of upper case letters.

If your word processor does not have an option for hardware input of lower case characters, you should select the software input option (if any). Every time you go into the editor, you should select the shift lock option OF THE WORD PROCESSOR. On some systems this is a double escape or perhaps a control C. Read your owner's manual to find out what the command is for your system. Once your word processor is in shift lock mode, it will probably read the keyboard directly without trying to convert anything to lower case. Since the Enhancer ][ produces true upper and lower case, you should be able to use the Enhancer ][, as you normally would, to enter upper and lower case characters. How long this shift lock will last (without having to re-issue the command to invoke it) varies from word processor to word processor. It may be that whenever you re-enter the editor, you will also have to reset the shift lock mode.

#### Section 7.a.2 A Note on Wordstar

The original release of Wordstar 3.0 had an option for hardware input of lower case. Unfortunately, this release contained some software errors dealing with the use of control K

## Chapter Seven: Other Software Considerations

and control A. MicroPro has indicated that a patch to correct this software problem will be available from MicroPro no later than November, 1981. If your version of Wordstar doesn't function properly, contact MicroPro directly.

Note: Wordstar is a registered trademark of MicroPro.

Apple CP/M is compatible with the Enhancer ][. Depending upon your system configuration, you may have to inform your CP/M system that you can display lower case characters.

If you have a Videoterm 80 column card, you will not need to perform the following operation. The normal Apple does not allow for the display of lower case characters in the 40 column mode, therefore, 40 column CP/M does not normally display lower case characters. Since the Lower Case Chip supplied with the Enhancer ][ gives your Apple lower case display capability, you will probably want to activate the lower case display ability of the CP/M system. To do this, you will need to perform the following:

A> MBASIC CONFIGIO

The A> is, of course, the CP/M prompt. You will then be asked if you have lower case, the answer is yes. Once you exit CONFIGIO, you will be able to display lower case letters.

### Section 7.c Other Commercial Software

How friendly your software packages are to lower case characters depends upon a number of independent variables. This section will discuss some of the reasons why certain programs work or don't work, how they might be made to work and the ramifications involved.

Generally speaking, the largest single factor in determining whether a program will like lower case characters is the environment in which it lives. On the Apple ][, we have three basic environments:

- i) Applesoft, Integer BASIC or DOS.
- ii) Pascal, FORTRAN, or any other language using the Pascal operating system.
- iii) CP/M

## Chapter Seven: Other Software Considerations

### Section 7.c.1 Applesoft, Integer BASIC and DOS.

Note: The following paragraph does not apply to systems using the Videoterm 80 column card.

Programs written in Applesoft, Integer BASIC or which use the input routines of the Apple ]['s monitor generally do not like lower case letters. This is usually due to CAPTST [section: 6.a; appendix: A] a routine of the Apple monitor. If CAPTST is the root of the problem, modifying CAPTST [appendix: A] might allow lower case entry. If CAPTST is not the problem, then the software package in question probably has its own input routines. If so, then the software itself will probably have to be modified. This is usually more work than profitable.

Assuming that one can get a program to accept lower case input, the battle is not necessarily over. Since upper case characters have completely different ASCII values than lower case characters, and since programs which are not designed to accept lower case input are not expecting lower case characters, it is likely that lower case characters cannot be substituted for their upper case equivalents.

Example: A program may stop and ask a yes or no question. You enter a lower case "y" in response. The program checks for an upper case "Y" only. If the response is not an upper case "Y", it assumes the answer was no. It didn't see an upper case "Y". You thought you said yes, the program thought that you said no.

### Section 7.c.2 Pascal, etc.

Note: FORTRAN and other languages which use the Pascal operating system have the same restrictions as Pascal.

Once you have made the SYSTEM.APPLE patch [section: X.f], you may enter lower case characters to your heart's content. Pascal recognizes lower case characters as legitimate commands. Again, the same problem as outlined in the example of section 7.c.1 above applies to programs written in Pascal as well.

Note: If you have a Videoterm 80 column card in slot 3, Pascal will turn it on when you boot. With the Videoterm, you do not need to make the patches to SYSTEM.APPLE as mentioned above.

## Chapter Seven: Other Software Considerations

### Section 7.c.3 CP/M

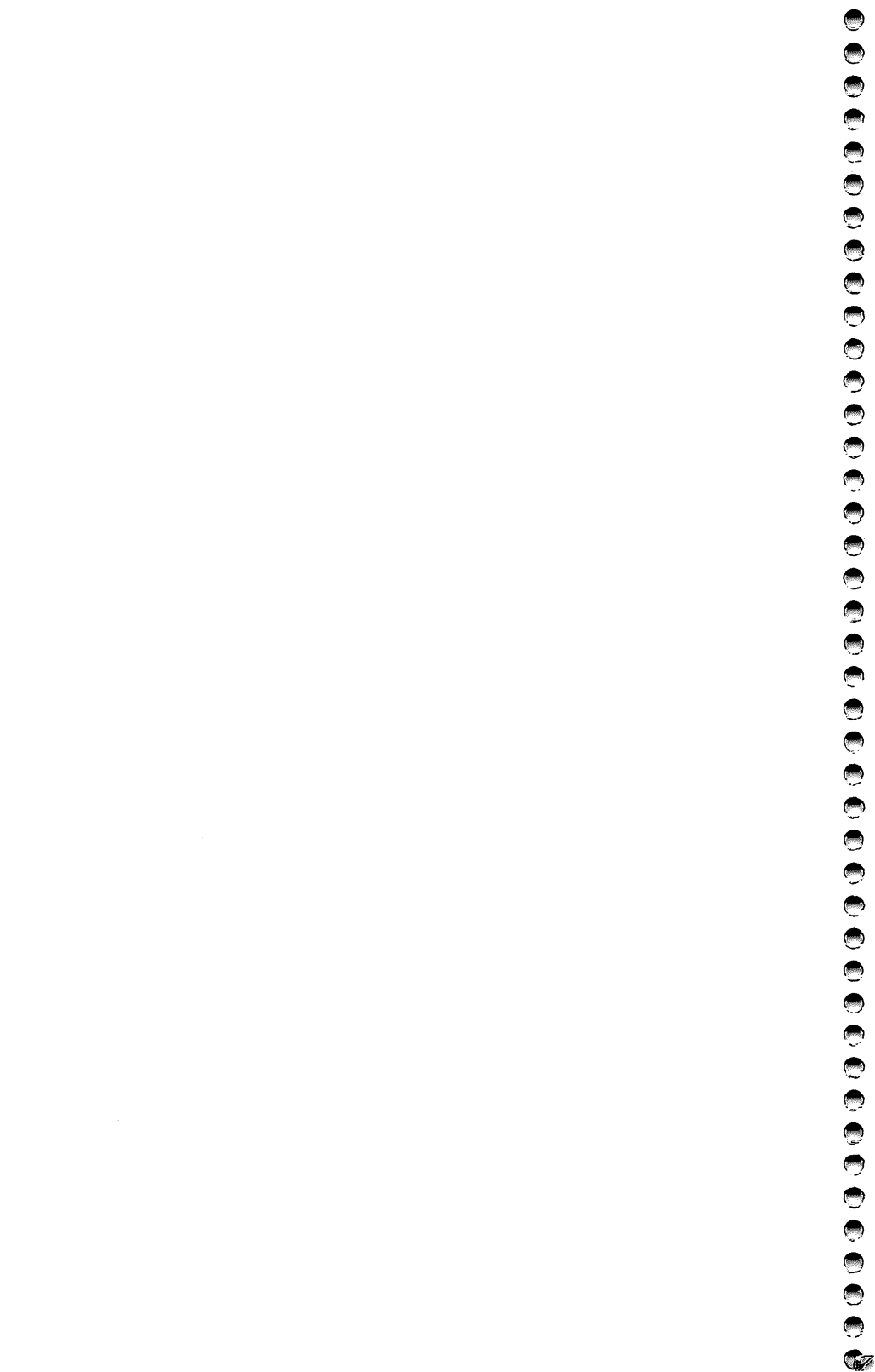
Any program in the CP/M or Softcard environment should be able to enter lower case letters directly from the keyboard. Naturally, CONFIGIO must be set for lower case display [section: 7.b]. Again, the restriction as illustrated in the example of section 7.c.1 applies to any CP/M based program.

Note: If you have a Videoterm 80 column card in slot 3, CP/M will turn it on when you boot. With the Videoterm, you do not need to run CONFIGIO as mentioned above.

## Chapter Eight: Peripheral Considerations

### Chapter Table of Contents.

- 8.a) Peripherals in General
- 8.b) Videoterm
- 8.c) Language Card and Other RAM Cards
- 8.d) Softcard (Z80)
- 8.e) Printers, Parallel and Serial I/O Boards
- 8.f) Disc Drive Controllers





## Chapter Eight: Peripherals Considerations

### Section 8.a Peripherals in General

There should be no interference between the Enhancer ][ and any peripheral. This is because the Enhancer ][ only modifies the output of the keyboard. It does not use any of the Apple's memory or other address lines. Even the wires that you installed for the type ahead buffer and the automatic down loading of the macro definitions from disc should not affect any peripheral. It is possible, however, that the software used to automatically down load the macro definitions could interfere with a device using the annunciator port number three. In this event, the Auto Down Load Disable option should be selected [section: 5.f.4].

Problems could arise if the firmware of some peripheral looked at the keyboard output for an upper case character and saw a lower case character instead. This problem may be negated by selecting Caps Lock mode.

### Section 8.b The Videoterm

The Videoterm 80 column card is completely compatible with the Enhancer ][. In fact, it can make life a lot easier since it does not try to convert lower case characters into upper case characters. The original Keyboard & Display Enhancer was originally conceived as an option for the Videoterm, but our engineers decided to make it a stand alone product which could be used in the 40 column mode as well.

Throughout this manual, you are likely to find notes indicating that use of the Videoterm 80 column card negates certain problems and negates the need for various patches.

### Section 8.c The Language Card and Other RAM Cards

There is no interference between the Enhancer ][ and any of the RAM cards available on the market. As a matter of fact, the use of a RAM card can actually be a benefit to the Enhancer ][ [section: A.d]. This is true in the 40 column Apple BASICS or DOS environments since the monitor may easily be modified to accept lower case input.

## Chapter Eight: Peripherals Considerations

### Section 8.d Softcard (Z80)

There is no problem using the Softcard in conjunction with the Enhancer ][. If you are using the Enhancer ][ without a Videoterm 80 column card under CP/M, you will probably want to refer to section 7.b.

### Section 8.e Printers, Parallel & Serial I/O Boards

There should be no interference between the Enhancer ][ and any I/O device. This is because the Enhancer ][ only modifies the output of the keyboard, it does not use any Apple ][ RAM or addresses other than the keyboard input byte. If you are having problems with your printer or other device, it is probably a software problem.

You should keep in mind that some software may be looking for upper case characters and does not expect the input of any lower case characters [section: 7.c.1]. Also, some printers cannot print lower case characters. To test your printer's capabilities, you might try the following from Applesoft:

```
]PR# <printer slot number>  
]FOR DU=32 TO 127:PRINT CHR$(DU) " ";:NEXT
```

Where the "]" is the Applesoft prompt (and is not typed into the computer). This should print all of the 96 printable ASCII characters that your printer can print. If you do not see any lower case characters printed, then your printer doesn't know how to print lower case characters.

### Section 8.f Disc Drive Controllers

Generally, the Enhancer ][ should be completely compatible with any disc drive and disc drive controller. Once again, the only problem that one is likely to encounter is that the Disc Operating System is likely not to be able to understand lower case characters [section: 7.c.1].

## Appendix A: The Lower Case Fix

While the Enhancer II turns your Apple ]['s keyboard into a full ASCII keyboard, not all programs will be able to read lower case letters and certain special symbols, i.e. ASCII characters with an ASCII value greater than \$E0 (224 decimal). This is because the routine CAPTST always subtracts \$20 (32 decimal) from ASCII characters greater than \$E0 (224 decimal). To input lower case characters directly from the keyboard, CAPTST must either be modified or bypassed. This appendix deals with some of the solutions which are possible.

Note: Use of the Videoterm 80 column board (or any environment other than Apple DOS or BASIC, e.g. Pascal, FORTRAN or CP/M) will bypass CAPTST.

There are basically four groups of system configuration:

- i) No card in slot zero (neither a RAM nor ROM card)
- ii) A ROM card in slot zero (either Applesoft or Integer BASIC)
- iii) An Apple RAM card (i.e. a Language Card) or other RAM card with an on board F8 ROM
- iv) A non-Apple RAM card (i.e. without the on board F8 ROM)

Note: This appendix explains how to create a binary file that can be used to program a 2716 EPROM. The 2716 may be plugged into any Apple Language Card. A 2716 EPROM is strongly recommended for those who are able to take advantage of this tact. A method of modifying a 2716 EPROM to plug directly into the on-board ROM sockets is also described. Extreme caution is urged when using a MODIFIED 2716 EPROM as damage to the computer or it's peripherals could result if it is improperly implemented.

### Section A.a CAPTST

CAPTST itself looks something like this:

```
FD7E: C9 E0      CAPTST      CMP #$E0      ;Greater than $E0 ?
FD80: 90 02      BCC FD84      ;No, Jump to next part
FD82: 29 DF      AND #$DF      ;Yes, Convert to CAPS
FD84: ...
```

While several fixes are possible, we suggest a change in the third instruction:

```
FD82: 29 FF      AND #$FF      ;Yes, Do nothing
```

## Appendix A: The Lower Case Fix

An AND with \$FF will, of course, do nothing at all. To do this, we need to change the byte at \$FD83 - not \$FD82 - from a \$DF to an \$FF.

### Section A.b Implementation

The Key Filter program [section: X.f] is a software method which will work for all system configurations, however it will not work with all programs. As with any program located in RAM, it is volatile and can be disconnected or destroyed. Other alternatives should be examined before deciding to rely on the Key Filter program. Likewise, the Key Filter program should be reviewed since it has some extra features. The Key Filter program is completely compatible with the methods of this appendix. Systems with ROM cards (either Applesoft or Integer) may program and modify a 2716 EPROM to replace the motherboard F8 ROM [section: A.c]. RAM card owners will find a number of solutions, depending upon their needs and exact system configuration [section: A.d].

### Section A.c The ROM Card Solution

Applesoft cards and Integer BASIC cards are the same except for the ROMs they come with. There is a single set of solder pads labeled F8 near the bottom of the card, toward the left of center. If this pad does not have a solder bridge across it, the F8 socket on the motherboard will always be active. Therefore, a modified 2716 EPROM [section: A.f] may be used in this socket. This solution will then allow direct entry of lower case characters in both Applesoft and Integer BASIC.

**Warning:** If a modified 2716 EPROM is placed in the F8 socket on the motherboard, care should be exercised at all times to ensure that the F8 solder pad is never bridged and that a RAM card is never used while the modified 2716 EPROM is in any socket on the motherboard. Failure to observe this restriction may result in permanent damage to your Apple ][ or any of it's peripherals. It is therefore suggested that labels reminding you of this be placed on the modified 2716 EPROM, the ROM Card and the RAM chip at socket location E3 on the motherboard.

**Note:** It is strongly suggested that the F8 ROM which would be replaced by the modified 2716 EPROM be kept and stored in conductive foam, a metal box or some aluminium foil to prevent damage from static electricity.

## Appendix A: The Lower Case Fix

### Section A.d The RAM Card Solution

There are a number of solutions available for RAM card systems. The optimal would be an Apple Language Card with your own 2716 EPROM on-board. This would allow easy implementation of the patches to CAPTST in both Applesoft and Integer BASIC. Things can become a little more complex if your RAM card does not have an onboard 2716 EPROM. Let us now examine both of these cases to see what can be done and how it may be done.

RAM cards duplicate the area of memory used by the on-board ROMs. This includes the F8 monitor ROM. Normally, when the resident language is being used, the ROMs on the motherboard are active and the RAM card is totally disabled. Likewise, when the non-resident language is being used, the RAM card is active and the motherboard ROMs are completely disabled.

Note: In the case of the Apple ][ Language Card, the motherboard F8 ROM is always disabled. It is always replaced by the ROM on-board the RAM card. That is, when the motherboard ROMs are active, the ROM on-board the RAM card is also active (replacing the disabled F8 ROM on the motherboard). When the RAM is active, all the ROMs are disabled (including the on-board ROM) and the F8 area in RAM is active.

### Section A.d.1 Apple ][ Language Cards

Since the on-board ROM completely replaces the motherboard F8 ROM, and since this ROM may be configured for a 2716 EPROM, one may make the changes necessary in the monitor and program a 2716 so that the resident language will have lower case capability. The non-resident language poses little problem since it is contained in RAM and stored on disc. It may therefore be easily modified and saved.

Note: If you do not intend to program a 2716 EPROM for use on your Apple ][ Language Card, you will not be able to modify the resident BASIC language (i.e. Applesoft on an Apple ][ plus). You should therefore refer to the procedure for RAM cards without the on-board F8 ROM.

#### Section A.d.1.1 Installing the 2716 EPROM

To use the programmed 2716 EPROM on the Language Card, you will need to configure the card for a 2716. If you examine the board, you will find a single pair of solder pads located near the top right corner labeled 2716. You should place a solder bridge across these pads, completing the circuit. Further examination should yield an etched symbol which looks something like two arrow heads meeting head on, or like the silhouette of an hour glass, near the bottom center of the board. The

## Appendix A: The Lower Case Fix

connection between the two triangles must be severed. Once both of these steps have been performed, the Language Card is configured for a 2716 EPROM.

The next step is to carefully replace the on-board ROM in the socket on the Language Card with the 2716 EPROM. The notched end of the EPROM should point toward the top of the board (as silk screened). You should ensure that the window portion of the EPROM has some kind of label over it to prevent program erasure.

### Section A.e How to Modify the Monitor Without a RAM Card

Modifications for Old Monitor ROM are NOT the same as those for the Autostart Monitor ROM. If you do not know which monitor ROM you have, perform this simple test. Turn your power off and turn it back on. If your disc drives start to boot or you see either the Applesoft or Integer BASIC prompt or you see APPLE ][ at the top of the screen, you have the Autostart Monitor, otherwise you should see a screen full of garbage with a monitor (asterisk) prompt at the lower left hand edge of the screen.

#### Section A.e.1 Modifying the Autostart Monitor

Boot DOS. From either BASIC, type: CALL -151 <cr> to get into the monitor. Now type the following:

```
800<F800.FFFFFM <cr>
800:4A 08 20 <cr>
BB3:C9 E0 B0 5 29 3F 9 40 60 29 1F 60 <cr>
D11:20 B3 FB EA <cr>
D83:FF <cr>
```

If you want a solid cursor, type: BBA:00 <cr>

Now type:

```
BSAVE MONITOR,A$800,L$800
```

This now completes the modifications to the monitor code.

## Appendix A: The Lower Case Fix

### Section A.e.2 Old Monitor ROM

Boot DOS. From either BASIC, type: CALL -151 <cr> to get into the monitor. Now type the following:

```
800<F800.FFFFM <cr>
D83:FF <cr>
```

If you want a solid cursor, type: D14:00 <cr>

Now type:

```
BSAVE MONITOR,A$800,L$800
```

This now completes the modifications to the monitor code.

### Section A.f Modifying a 2716 EPROM

Warning: The following modification should never be used in conjunction with a RAM card. Doing so could cause damage to your RAM card.

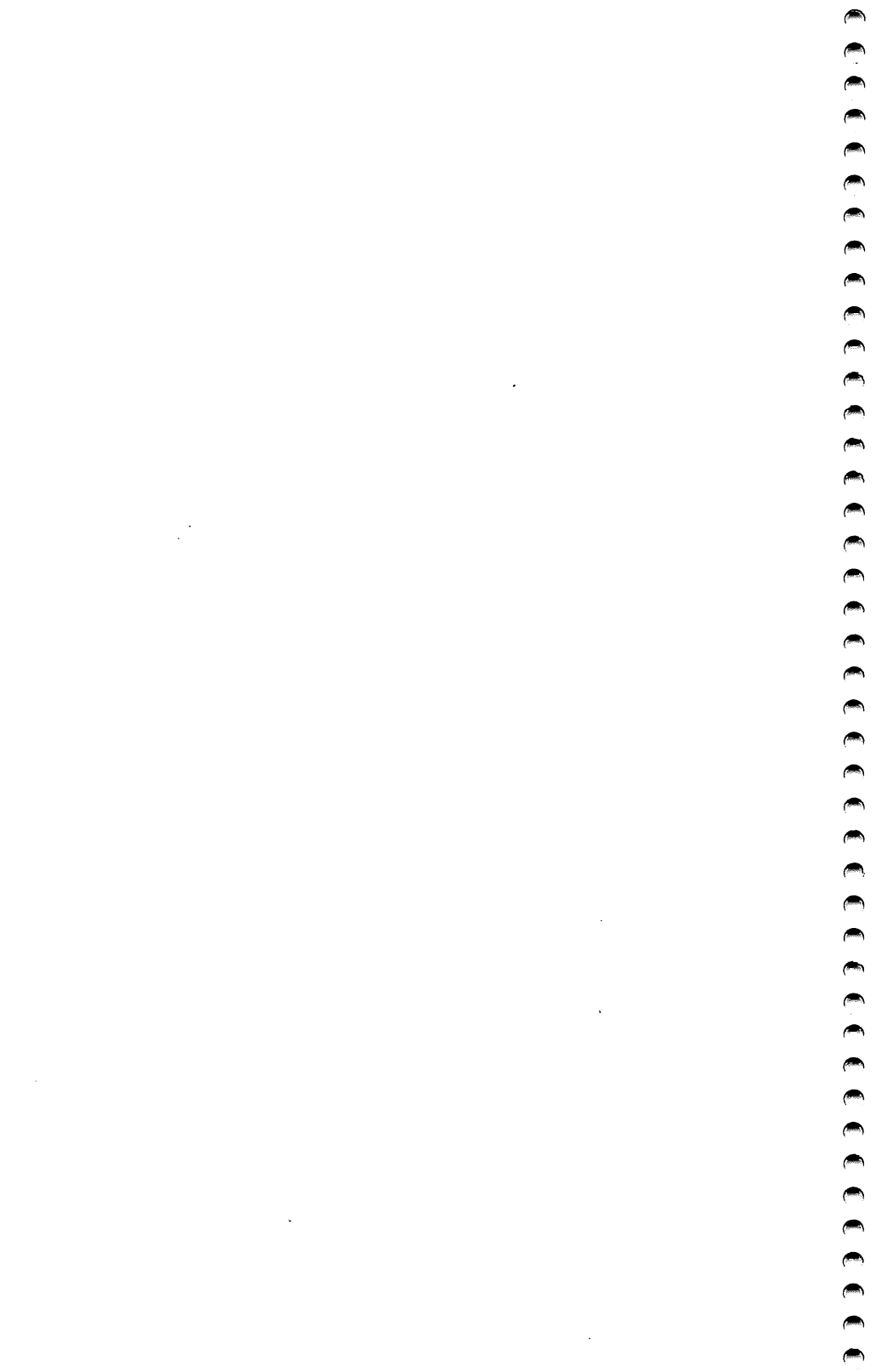
Note: Apple ][ Language Cards can be configured for unmodified 2716 EPROMs.

To allow the 2716 to be plugged directly into a ROM socket on the Apple ][ motherboard (i.e. a socket which CANNOT be configured for a 2716 EPROM), you will need to make the following modifications to the EPROM AFTER it has been burned:

Place a jumper wire between pins 12 and 18. Place another jumper wire between pins 21 and 24. These jumper wires should be soldered into place. Bend pins 18 and 21 up flat next to the bottom of the chip so that the pins do not go into the motherboard socket.

Naturally, you will want to place some kind of label over the window portion of the chip to prevent it from being erased.

Note: The pins are numbered from 1 to 24 on a 2716. One end of the chip will have a notch, dot or depression. Placing this at the twelve o'clock position with the top (window) side up, pin one is the first pin to the left (counter-clockwise). The pins are numbered counter-clockwise around the chip. Pin 24 is directly opposite pin 1.





## Appendix B: Lower Case Display

### Section B.a Revision 0 through 6 Apples

The Enhancer ][ may be used on revision 0 through 6 Apples with certain restrictions. Since the Enhancer ][ replaces the keyboard encoder board, the keyboard must be the piggyback style. Though revision 0 through 6 Apples usually don't have this style of keyboard, the user may purchase one from Apple - WITHOUT the encoder board - and install it on their system. Older style keyboards may be used on the newer machines and vice versa.

The other problem that one faces is that of lower case display. The Lower Case Chip CANNOT be used with revision 0 through 6 Apples. Here are some solutions:

- i) Use the Videoterm 80 column card. It is capable of displaying lower case characters in the 80 column mode.
- ii) Modify the an old Keyboard & Display Enhancer I to be a display only device [section: B.b].
- iii) Use some other revision 0 through 6 Apple 40 column lower case display device

Note: This manual assumes that you have a revision 7 or greater Apple. If you have a revision 0 through 6 Apple, all references to lower case display (in 40 columns) does not apply unless you have implemented options ii or iii above.

### Appendix B.b How to Modify Your Old Enhancer for Display Only

Note: This section is applicable only to Revision 0 through 6 motherboards only.

This section addresses a modification which can be made to the original Keyboard & Display Enhancer to turn it into a display only device. This will allow the Keyboard & Display Enhancer the be used as a 40 column lower case character generator and the Enhancer ][ to be used as normal. The Enhancer ][ will, of course, have to be installed on a piggyback style keyboard.

All of these modifications may be made on the back side of the Keyboard & Display Enhancer. The first step is to disable the control and shift key inputs. To do this we need only one piece of wire not much longer than 6 centimeters (approximately 2 inches). This may be a bare wire of just about any thickness. Wire wrap wire will do just fine. Referring to page A-4 of the Keyboard & Display Enhancer Owner's Reference Manual, you should see the 6 fingered Molex connector jack to the right of the

## Appendix B: Lower Case Display

board in the photograph. Pins 1 and 3 must both be connected to ground (i.e. Pin 7 of U6). You may connect all six pins to ground, if you like.

The next step is to jumper the normally no-connections from the keyboard socket to the keyboard header going to the A7 socket on the motherboard. There are only two that we will need. One is for the type ahead buffer, the other for the automatic down loading of macro definitions. Simply jumper Pin 9 of P3 to Pin 9 of J1 and Pin 4 of P3 to Pin 4 of J1.

Note: You should place a piece of paper inbetween the two Enhancers to prevent electrical contact. A standard piece of note book paper will probably due the job nicely.

## Appendix C: Down Load Technical Data

The data transmitted to the keyboard during down load is of the following format:

| byte                  | name   | function   |
|-----------------------|--------|--|
| 0                     | status | Selects down load options                        |
| 1 & 2                 | pntend | Pointer to end of table<br>(Low byte, high byte) |
| All<br>other<br>bytes | macdef | Macro definition (format below)                  |
| Last<br>Byte          | tabend | End of Table (must be zero)                      |

Where STATUS is of the following format:

|     |      |  |
|-----|------|--|
| 0   | \$0  | Bit 0: Disable Shift Lock                        |
| 2   | \$2  | Bit 1: Lock Keyboard Mode                        |
| 4   | \$4  | Bit 2: Select Lower Case Mode after Down Load    |
| 8   | \$8  | Bit 3: Disable Auto Repeat                       |
| 16  | \$10 | Bit 4: Disable Type Ahead                        |
| 32  | \$20 | Bit 5: Disable Macro Entry & Clear from Keyboard |
| 64  | \$40 | Bit 6: Lock Out Auto Down Load                   |
| 128 | \$80 | Bit 7: Unused                                    |

If any given bit is on, that option is selected. If Bit 2 is off, Caps Lock Mode is selected after down load. If Bit 6 is on, down load is disabled and it is impossible to change any of the above conditions until the system has been powered off.

PNTEND is equal to the number of bytes in MACDEF + \$201.

Where MACDEF (macro definition) is in the following format:

| byte                  | name  | function   |
|-----------------------|-------|--|
| 0                     | mcsib | Contains information on mode, control & shift keys |
| 1                     | matrx | Location on key map                                |
| All<br>other<br>bytes | macro | ASCII values of macro definition, high bit set     |

## Appendix C: Down Load Technical Data

MCSIB is a value from 0 to 7. If MCSIB is in the range of 0 to 3, the macro being defined is in Caps Lock Mode. If MCSIB is in the range of 4 to 7, the macro is defined in Lower Case Mode. The following table defines MCSIB:

|                 | Caps Lock Mode | Lower Case Mode |
|-----------------|----------------|-----------------|
| Alone           | 0              | 4               |
| Control         | 1              | 5               |
| Shift           | 2              | 6               |
| Control - Shift | 3              | 7               |

Matrx is the location in the keyboard matrix. This table lists the values in both decimal and hexadecimal:

| Decimal:  | 0             | 16        | 32   | 48     |
|-----------|---------------|-----------|------|--------|
| Hex: \$00 | \$10          | \$20      | \$30 |        |
| 0 \$0     | 3 U           | C         |      |        |
| 1 \$1     | 4 I           | V         |      | Return |
| 2 \$2     | 5 O           | B         |      |        |
| 3 \$3     | 6 P           | N         |      |        |
| 4 \$4     | 7 D           | M         |      |        |
| 5 \$5     | 8 F           | ,         |      |        |
| 6 \$6     | 9 G           | .         |      |        |
| 7 \$7     | 0 H           | /         |      |        |
| 8 \$8     | : J           | S         |      |        |
| 9 \$9     | - K           | 2         |      |        |
| 10 \$A    | Q L           | 1         |      |        |
| 11 \$B    | W ;           | Escape    |      |        |
| 12 \$C    | E Left Arrow  | A         |      |        |
| 13 \$D    | R Right Arrow | Space Bar |      |        |
| 14 \$E    | T Z           |           |      |        |
| 15 \$F    | Y X           |           |      |        |

## Appendix V: Tech's Installation Checklist

These are the tools required to install the Enhancer ][:

- i) Phillips screwdriver
- ii) Pliers (preferably needle-nose)
- iii) Wire cutters
- iv) Wire stripper

- ( ) Unplug the Apple and remove all the cards from the slots.
- ( ) Remove the case from the bottom of the Apple.
- ( ) Replace the chip in socket A-5 with the one marked 'Lower Case'.
- ( ) Remove the piggyback board from the keyboard.
- ( ) Transfer the 16 pin cable to the equivalent socket on the Enhancer ][.
- ( ) Now, look at the underside of the keyboard; On some Apples there will be a metal stiffener bar across the back of the keyboard; a strip of insulating material should be placed over the edge of this bar so that when the Enhancer ][ is installed, no bare metal touches it.
- ( ) Look now at the plastic spacers that held the piggyback board on. The right spacer (the one further from the side of the Apple) must be rotated 90 degrees so its flanges are parallel to the edge of the keyboard [photo: 2.13].
- ( ) Install the Enhancer ][ on the keyboard in place of the piggyback board you removed.
- ( ) Optional: Install a wire leading from pin 5 of socket B-7 to pin 9 of the keyboard cable socket. This enables the type-ahead buffer.
- ( ) Optional: Install another wire leading from pin 12 of socket F-14 to pin 4 of the keyboard cable socket and REMOVE THE BLUE SHORTING PLUG FROM PIN 5 OF THE MOLEX CONNECTOR [photo: 2.11]. This enables totally automatic downloading of key redefinitions.
- ( ) Place the case of the Apple back on its base, being careful not to crush the keyboard cable plug.

## Appendix V: Tech's Installation Checklist

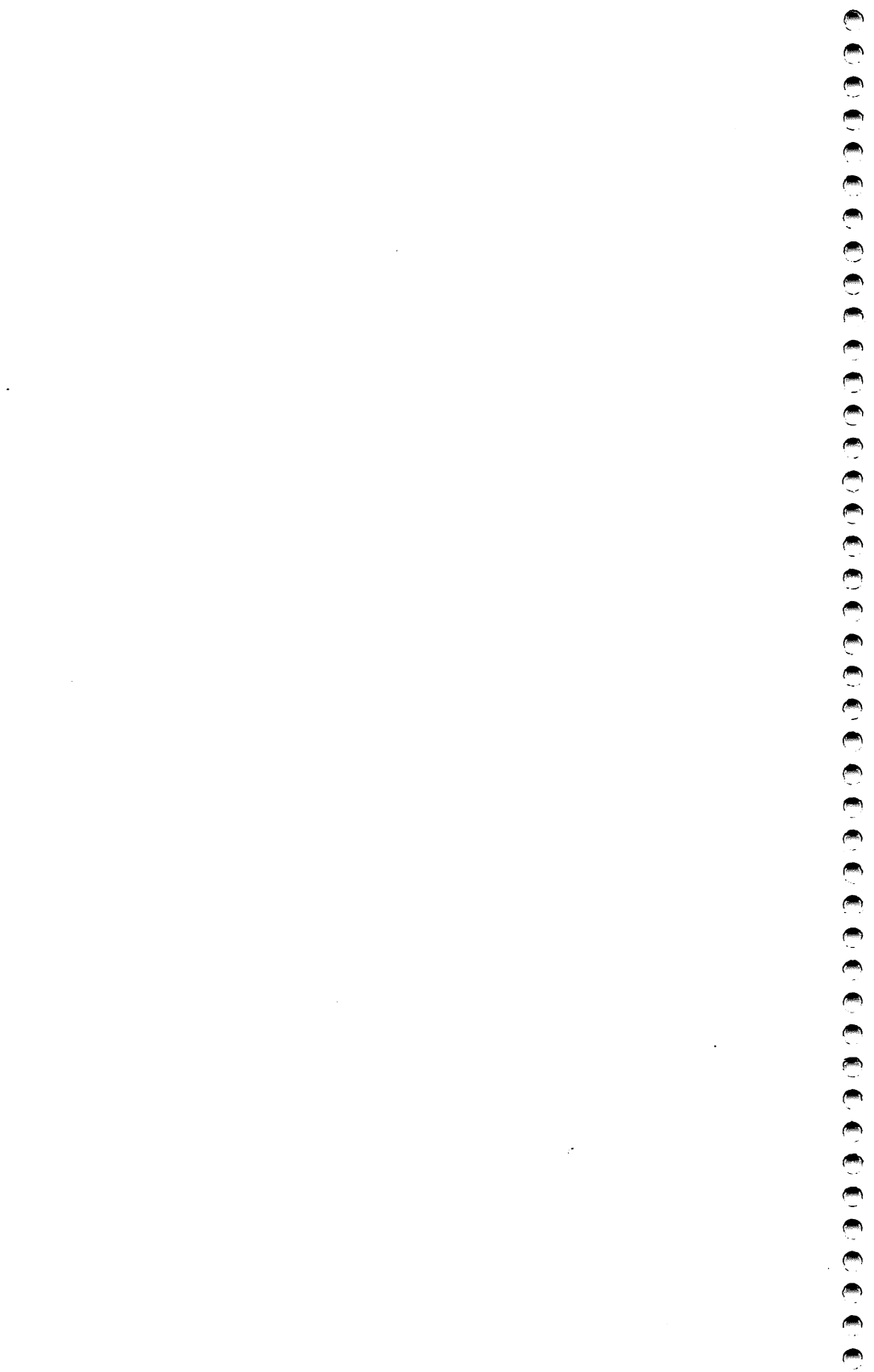
- ( ) Plug the keyboard cable connector into its socket, making sure that pin one of the cable enters hole number one of the socket.
- ( ) Reinstall two of the screws immediately under the keyboard, but don't bother replacing the washers.
- ( ) Proceed to the installation checkout section. If everything checks out ok, complete the re-assembly of the Apple ][ (with the power off), else refer to Chapter 3.

End of installation.

## Appendix W: Specifications

Microprocessor: 6504 (uses the 6502 instruction set)  
RAM: 1K static (low power)  
Firmware ROM: 2716 EPROM  
Dimensions: 16.1 cm by 7.4 cm  
Technology: LS TTL, MOS, and CMOS  
Type ahead buffer: 128 characters  
Auto repeat: approximately 15 characters per second  
Fast repeat: approximately 50 characters per second  
Installation: replaces keyboard encoder board  
Control - Reset protection: plug/jumper selectable  
Key redefinitions: down loadable from disc  
Memory for macro definitions: 1/2 K (up to 170 single  
character macros)  
Self test diagnostics: RAM test & ROM checksum,  
automatic on power up.

\* Specifications are subject to change without notice.





## Appendix X: Supporting Software

### Section X.a The Configuration & Hello Programs

The Configuration & Hello Programs run, for the most part, automatically. When run the first time, the Hello program will call one of the configuration programs. The configuration programs request that you insert a DOS system master diskette. This is to load in the FPBASIC and INTBASIC files, modify them [appendix: A] with the Lower Case Fix and save them on the Enhancer ][ Utilities Disc.

The configuration program will ask you whether you want to load Integer BASIC or Applesoft in the RAM card. You should respond with the BASIC that you plan to use the most. (If you have a RAM card with an on-board F8 ROM and you have burned your own 2716 EPROM monitor ROM, then you should respond with the BASIC that is not resident.)

There are two Hello programs on the disc. One is in Applesoft, the other in Integer BASIC. The Applesoft program is named HELLO. The DOS is set to boot to this file. The Integer file is named APPLESOFT. If you boot this disc on an Integer system, DOS will attempt to load in HELLO, but since it is in Applesoft, it will attempt to load Applesoft into the system from a file named APPLESOFT. Therefore, regardless of what kind of ROMs you have, the appropriate file will be run upon boot-up.

Line 10 of the Hello program controls what the program will do when booted. If it is missing, the configuration program is called. Otherwise, line 10 will set the variable I equal to 1, 2 or 3. If it is 1, Key Filter is run. If it is 2, Integer BASIC is loaded into the language card. If it is 3, Applesoft is selected (to be loaded into the language card).

Note: The configuration programs produce super fast loading FPBASIC and INTBASIC files.

## Appendix X: Supporting Software

]LIST

```
5 POKE 216,0: TEXT : PRINT : HOME : POKE - 23112,129: POKE - 23104,128
   : REM CHANGE DOS TO SELECT NON RESIDENT BASIC FIRST
15 PRINT "DOS VERSION 3.3": IF LEN (J$) THEN PRINT CHR$ (4)"BRUN"J$
20 PRINT : PRINT "Apple II plus

                                     Loading ";

25 ON I GOTO 50,100,150
30 PRINT "Configuration program"
40 PRINT CHR$ (4)"RUN CONFIGURE.A"
50 PRINT "Keyfilter"
60 PRINT CHR$ (4)"BRUN KEYFILTER"
70 PRINT CHR$ (4)"FP"
100 PRINT "Integer into Language card"
120 PRINT CHR$ (4)"BRUN INTBASIC"
140 END
150 PRINT "Applesoft into Language card"
170 PRINT CHR$ (4)"BRUN FPBASIC"
180 POKE 49280,0
190 END
```

]

## Appendix X: Supporting Software

```

10 I = 2
20 REM
30 POKE 768,0: POKE 769,173: POKE 770,0: POKE 771,224: POKE 772,72: POKE
  773,173: POKE 774,129: POKE 775,192: POKE 776,104:
40 POKE 777,72: POKE 778,205: POKE 779,0: POKE 780,224: POKE 781,208: POKE
  782,35: POKE 783,173: POKE 784,131: POKE 785,192:
50 POKE 786,173: POKE 787,131: POKE 788,192: POKE 789,169: POKE 790,165:
  POKE 791,141: POKE 792,0: POKE 793,208: POKE 794,205:
60 POKE 795,0: POKE 796,208: POKE 797,208: POKE 798,19: POKE 799,74: POKE
  800,141: POKE 801,0: POKE 802,208: POKE 803,205:
70 POKE 804,0: POKE 805,208: POKE 806,208: POKE 807,10: POKE 808,173: POKE
  809,129: POKE 810,192: POKE 811,173: POKE 812,129:
80 POKE 813,192: POKE 814,169: POKE 815,1: POKE 816,208: POKE 817,2: POKE
  818,169: POKE 819,0: POKE 820,141: POKE 821,0:
90 POKE 822,3: POKE 823,104: POKE 824,205: POKE 825,0: POKE 826,224: POKE
  827,240: POKE 828,3: POKE 829,173: POKE 830,128:
100 POKE 831,192: POKE 832,96:
110 CALL 769
120 IF PEEK (768) < > 1 THEN I = 1: GOTO 280
130 REM
140 PRINT : PRINT "Language card found"
150 PRINT : INPUT "Insert DOS 3.3 Master diskette ";A$
160 PRINT CHR$ (4)"BLOAD INTBASIC,A$2000": PRINT CHR$ (4)"BLOAD FPBASI
  C,A$5000"
170 POKE 19379,201: POKE 19380,224: POKE 19381,176: POKE 19382,5: POKE 1
  9383,41: POKE 19384,63: POKE 19385,9: POKE 19386,64
180 POKE 19387,96: POKE 19388,41: POKE 19389,31: POKE 19390,96: POKE 197
  29,32: POKE 19730,179: POKE 19731,251: POKE 19732,234: POKE 19843,25
  5
190 POKE 31667,201: POKE 31668,224: POKE 31669,176: POKE 31670,5: POKE 3
  1671,41: POKE 31672,63: POKE 31673,9: POKE 31674,64
200 POKE 31675,96: POKE 31676,41: POKE 31677,31: POKE 31678,96: POKE 320
  17,32: POKE 32018,179: POKE 32019,251: POKE 32020,234: POKE 32131,25
  5
210 PRINT : INPUT "Insert Enhancer Utility diskette ";A$
220 PRINT CHR$ (4)"BLOAD BASIC": PRINT CHR$ (4)"BSAVE INTBASIC,A$1F04,
  L$30FB": PRINT CHR$ (4)"BSAVE INTBASIC,A768,L108": PRINT CHR$ (4)"
  BSAVE FPBASIC,A$4F04,L$30FB": PRINT CHR$ (4)"BSAVE FPBASIC,A768,L10
  8"
230 GOTO 500
300 PRINT : PRINT : PRINT : PRINT : PRINT CHR$ (4)"OPEN CONFIG"
310 PRINT CHR$ (4)"WRITE CONFIG"
320 PRINT "LOAD HELLO"
330 PRINT "10 I="I":J$=" CHR$ (34)A$ CHR$ (34)
340 PRINT "SAVE HELLO"
350 PRINT "RUN"
360 PRINT CHR$ (4)"CLOSE"
370 PRINT CHR$ (4)"EXEC CONFIG"
380 END
500 PRINT : PRINT : PRINT "Select one:"
510 PRINT : PRINT "1 Load Integer into Language card"
520 PRINT : PRINT "2 Load Applesoft into Language card"
530 VTAB PEEK (37) - 4: HTAB 12
540 GET A$:I = VAL (A$) + 1: IF I < > 2 AND I < > 3 THEN 541
550 VTAB PEEK (37): HTAB 1: INPUT "Enter Macro file to download
  or RETURN for none ";A$
560 GOTO 300

```

## Appendix X: Supporting Software

### Section X.b The Key Filter Program

There is a program on the Enhancer ][ Utilities Disc called Key Filter. Key Filter is a software method of bypassing CAPTST. It can be used to allow input of lower case characters by most programs written in either BASIC or 6502 machine language. The Key Filter program should not be used with word processors [section: 7.a].

Since the Key Filter program lives in RAM, it will take up some of your free memory - though not a great deal. It is vulnerable to attack from other programs. It can be disconnected with certain monitor, DOS and BASIC commands. It must be loaded from disc each time the power is turned on or DOS is booted. Key Filter is a good solution for many people, however, there are other solutions which we shall discuss in subsequent sections.

#### Section X.b.1.1 Installing Key Filter

The disc which accompanies your Enhancer ][ contains a copyable version of Key Filter. You may relocate the Key Filter program by running the program FID on your DOS 3.3 system master disc or by inserting the Enhancer ][ Utilities Disc in one of your disc drives and issue the following DOS command:

```
BLOAD KEY FILTER,A$800
```

To save it onto another disc, place the destination disc into your disc drive and type:

```
BSAVE KEY FILTER,A$800,L$<unknown>
```

*A\$60007L\$01F0*

If you wish to save Key Filter on more than one disc, you may repeat the second step above without having to reload Key Filter from the Enhancer ][ Utilities Disc.

#### Section X.b.1.2 Using Key Filter

To use Key Filter, the DOS commands:

```
BRUN KEY FILTER  
FP or INT
```

should be issued. This will cause Key Filter to be loaded into memory and set up all the necessary vectors.

## Appendix X: Supporting Software

Note: The Hello program on the Enhancer ][ Utilities Disc will configure itself to automatically load in Key Filter if you do not have a language card. If you do have a language card but wish to have Key Filter loaded in each time you boot, change line ten of the Hello program as follows:

```
10 I = 1
```

After you make this change, you should save the program.

Note: If you have an Apple ][ plus, the name of your Hello program should be: HELLO. If you have Integer BASIC in ROM (non-plus systems), the name of your Hello program should be: APPLESOFT.

Note: If you want to reconfigure the HELLO program, remove line ten from the program by typing:

```
10 <cr>
```

Issuance of the PR#n1 and IN#n2 commands, or the equivalent monitor commands, where n1 and n2 are integer values between 0 and 7 inclusive (e.g. PR#0:IN#0) will disconnect Key Filter. If Key Filter becomes disconnected, it may be reconnected with a control reset (Autostart ROM only), by a & command from Applesoft, a 3F5G from the monitor or by a CALL 1013 from either BASIC.

Example:

```
*3F5G
or
] &
or
>CALL 1013
```

```

6 *****
7 *
8 *          ENHANCER INTERFACE          *
9 *
10 *          SOFTWARE 2.0                *
11 *
12 *          10/31/1981          11:30    *
13 *
14 *****
15 *
16 *
17 BASL      EQU  $28
18 PROMPT    EQU  $33
19 CSWL      EQU  $36
20 KSWL      EQU  $38
21 A1L       EQU  $3C
22 A1H       EQU  $3D
23 A2L       EQU  $3E
24 A2H       EQU  $3F
25 A4L       EQU  $42
26 A4H       EQU  $43
27 IN        EQU  $200
28 RSVECL    EQU  $3F2
29 RSVECH    EQU  $3F3
30 AMPER     EQU  $3F5
31 DEST      EQU  $9B00
32 KEYIN     EQU  $FD1B
33 COUT1     EQU  $FDF0
34 MOVE      EQU  $FE2C
35 *
36          OBJ  $6000
37          ORG  $6000
38 *
39 *****
40 *
41 *          RELOCATE KEYFILTER          *
42 *
43 *****
44 *
45 RELOCATE LDA  #$9A
6002: 8D 01 9D 46          STA  $9D01
6005: A9 00 47          LDA  #<DEST
6007: 85 42 48          STA  A4L
6009: A9 9B 49          LDA  #>DEST
600B: 85 43 50          STA  A4H
600D: A9 25 51          LDA  #<START
600F: 85 3C 52          STA  A1L
6011: A9 60 53          LDA  #>START
6013: 85 3D 54          STA  A1H
6015: A9 4F 55          LDA  #<END
6017: 85 3E 56          STA  A2L
6019: A9 61 57          LDA  #>END
601B: 85 3F 58          STA  A2H
601D: A0 00 59          LDY  #$00
601F: 20 2C FE 60        JSR  MOVE
6022: 4C 00 9B 61        JMP  DEST
62 *
63 *****

```

```

64 *
65 *          START UP ROUTINE          *
66 *                                     *
67 *****
68 *
69 START      ORG  DEST
70 *
9B00: 20 2A 9B 71  STARTUP  JSR  INITIAL
9B03: A9 4C 72          LDA  #$4C          ; ESTABLISH & AND RESET VECTORS
9B05: 8D F5 03 73          STA  AMPER
9B08: A9 00 74          LDA  #<STARTUP
9B0A: 8D F6 03 75          STA  AMPER+1
9B0D: A9 9B 76          LDA  #>STARTUP
9B0F: 8D F7 03 77          STA  AMPER+2
9B12: A9 24 78          LDA  #<RESTART
9B14: 8D F2 03 79          STA  RSVECL
9B17: A9 9B 80          LDA  #>RESTART
9B19: 8D F3 03 81          STA  RSVECH
9B1C: 49 A5 82          EOR  #$A5
9B1E: 8D F4 03 83          STA  RSVECH+1
9B21: 4C EA 03 84          JMP  $3EA
85 *
9B24: 20 2A 9B 86  RESTART  JSR  INITIAL
9B27: 4C D0 03 87          JMP  $3D0
88 *
89 *****
90 *                                     *
91 *          INITIALIZATION          *
92 *                                     *
93 *****
94 *
9B2A: A9 4C 95  INITIAL  LDA  #<KEYINI      ; CHANGE INPUT AND OUTPUT HOOKS
9B2C: 85 38 96          STA  KSWL
9B2E: A9 9B 97          LDA  #>KEYINI
9B30: 85 39 98          STA  KSWL+1
9B32: A9 0B 99          LDA  #<COUT2
9B34: 85 36 100         STA  CSWL
9B36: A9 9C 101         LDA  #>COUT2
9B38: 85 37 102         STA  CSWL+1
9B3A: A9 00 103         LDA  #$00          ; INITIALIZE VARIABLES
9B3C: 8D 24 9C 104         STA  XSAVE
9B3F: 8D 25 9C 105         STA  YSAVE
9B42: 8D 26 9C 106         STA  OLDCHAR
9B45: 8D 27 9C 107         STA  FLAGS
9B48: 8D 28 9C 108         STA  ESCFLG
9B4B: 60 109          RTS
110 *
111 *****
112 *                                     *
113 *          INPUT ENTRY POINT          *
114 *                                     *
115 *****
116 *
9B4C: 8E 24 9C 117  KEYINI  STX  XSAVE          ; SAVE X
9B4F: 48 118          PHA
9B50: C9 E0 119          CMP  #$E0          ; FIX LOWER CASE CURSOR
9B52: 90 04 120          BLT  SKIP1
9B54: 29 1F 121          AND  #$1F

```

|       |    |    |     |         |      |          |                                       |
|-------|----|----|-----|---------|------|----------|---------------------------------------|
| 9B56: | 91 | 28 | 122 |         | STA  | (BASL),Y |                                       |
| 9B58: | E0 | 00 | 123 | SKIPI   | CPX  | #\$00    |                                       |
| 9B5A: | F0 | 1A | 124 |         | BEQ  | GETLN    | ; IF ZERO ASSUME GETLN                |
| 9B5C: | CA |    | 125 |         | DEX  |          |                                       |
| 9B5D: | AD | 26 | 9C  | 126     | LDA  | OLDCHAR  | ; GET LAST CHARACTER FROM GETLN       |
| 9B60: | C9 | 88 | 127 |         | CMP  | #\$88    | ; IF BS ASSUME GETLN                  |
| 9B62: | F0 | 12 | 128 |         | BEQ  | GETLN    |                                       |
| 9B64: | DD | 00 | 02  | 129     | CMP  | IN,X     |                                       |
| 9B67: | F0 | 0D | 130 |         | BEQ  | GETLN    |                                       |
| 9B69: | 29 | DF | 131 |         | AND  | #\$DF    |                                       |
| 9B6B: | DD | 00 | 02  | 132     | CMP  | IN,X     | ; IF SAME AS CHARACTER IN INPUT       |
| 9B6E: | D0 | 28 | 133 |         | BNE  | NTGETLN  | ; BUFFER THEN ASSUME GETLN            |
| 9B70: | AD | 26 | 9C  | 134     | LDA  | OLDCHAR  | ; GET LAST CHARACTER FROM GETLN       |
| 9B73: | 9D | 00 | 02  | 135     | STA  | IN,X     | ; FIX INPUT BUFFER                    |
| 9B76: | 38 |    | 136 | GETLN   | SEC  |          | ; SET GETLN FLAG                      |
| 9B77: | 6E | 27 | 9C  | 137     | ROR  | FLAGS    |                                       |
| 9B7A: | 68 |    | 138 |         | PLA  |          |                                       |
| 9B7B: | AE | 24 | 9C  | 139     | LDX  | XSAVE    |                                       |
| 9B7E: | 20 | 1B | FD  | 140     | JSR  | KEYIN    | ; GET CHARACTER FROM KEYBOARD         |
| 9B81: | 2C | 28 | 9C  | 141     | BIT  | ESCFLG   |                                       |
| 9B84: | 30 | 2C | 142 |         | BMI  | ESC1     | ; IF LAST KEY WAS ESC THEN FINISH IT  |
| UP    |    |    |     |         |      |          |                                       |
| 9B86: | C9 | 95 | 143 |         | CMP  | #\$95    | ; CHECK FOR COPY KEY                  |
| 9B88: | D0 | 02 | 144 |         | BNE  | NOTPICK  |                                       |
| 9B8A: | B1 | 28 | 145 |         | LDA  | (BASL),Y | ; GET CHARACTER FROM SCREEN           |
| 9B8C: | C9 | 9B | 146 | NOTPICK | CMP  | #\$9B    | ; CHECK FO ESCAPE                     |
| 9B8E: | F0 | 17 | 147 |         | BEQ  | ESC      | ; BEGIN ESC SEQUENCE                  |
| 9B90: | 8D | 26 | 9C  | 148     | STA  | OLDCHAR  |                                       |
| 9B93: | C9 | 8D | 149 |         | CMP  | #\$8D    | ; IF CR THEN FIX INPUT BUFFER         |
| 9B95: | F0 | 3E | 150 |         | BEQ  | FIXBUFF  |                                       |
| 9B97: | 60 |    | 151 |         | RTS  |          |                                       |
| 9B98: | 68 |    | 152 | NTGETLN | PLA  |          |                                       |
| 9B99: | AE | 24 | 9C  | 153     | LDX  | XSAVE    |                                       |
| 9B9C: | 20 | 1B | FD  | 154     | JSR  | KEYIN    | ; GET CHARACTER FROM KEYBOARD         |
| 9B9F: | 48 |    | 155 |         | PHA  |          |                                       |
| 9BA0: | A9 | 00 | 156 |         | LDA  | #\$00    |                                       |
| 9BA2: | 8D | 26 | 9C  | 157     | STA  | OLDCHAR  |                                       |
| 9BA5: | 68 |    | 158 |         | PLA  |          |                                       |
| 9BA6: | 60 |    | 159 |         | RTS  |          |                                       |
|       |    |    | 160 | *       |      |          |                                       |
| 9BA7: | 48 |    | 161 | ESC     | PHA  |          |                                       |
| 9BA8: | A9 | 88 | 162 |         | LDA  | #\$88    |                                       |
| 9BAA: | 8D | 28 | 9C  | 163     | STA  | ESCFLG   | ; SET ESCAPE FLAG                     |
| 9BAD: | 8D | 26 | 9C  | 164     | STA  | OLDCHAR  | ; AND OLDCHAR                         |
| 9BB0: | 68 |    | 165 |         | PLA  |          |                                       |
| 9BB1: | 60 |    | 166 |         | RTS  |          |                                       |
|       |    |    | 167 | *       |      |          |                                       |
| 9BB2: | 8C | 25 | 9C  | 168     | ESC1 | STY      | YSAVE ; SAVE Y                        |
| 9BB5: | C9 | E0 | 169 |         | CMP  | #\$E0    | ; MAKE UPPER CASE FOR ESCAPE FUNCTION |
| N     |    |    |     |         |      |          |                                       |
| 9BB7: | 90 | 02 | 170 |         | BLT  | SKIP     |                                       |
| 9BB9: | 29 | DF | 171 |         | AND  | #\$DF    |                                       |
| 9BBB: | A0 | 03 | 172 | SKIP    | LDY  | #\$03    | ; CHECK FOR TYPE OF ESCAPE FUNCTION   |
| 9BED: | D9 | D1 | 9B  | 173     | LOOP | CMP      | XTBL,Y                                |
| 9BC0: | F0 | 06 | 174 |         | BEQ  | ESC2     |                                       |
| 9BC2: | 88 |    | 175 |         | DEY  |          |                                       |
| 9BC3: | 10 | F8 | 176 |         | BPL  | LOOP     |                                       |
| 9BC5: | 4E | 28 | 9C  | 177     | LSR  | ESCFLG   |                                       |



```

9BC8: A0 88      178 ESC2    LDY  #$88
9BCA: 8C 26 9C   179        STY  OLDCHAR
9BCD: AC 25 9C   180        LDY  YSAVE
9BD0: 60          181        RTS
9BD1: C9 CA CB
9BD4: CD          182 XTBL    HEX  C9CACBCD
          183 *
          184 *
          185
9BD5: 48          185 FIXBUFF PHA                ; CONVERT LOWER CASE TO UPPER CASE
9BD6: A5 33      186        LDA  PROMPT          ; EXCEPT THOSE WITHIN QUOTES
9BD8: C9 BE      187        CMP  #">"          ; OR INPUTS WITHOUT > OR ] PROMPTS
9BDA: FO 04      188        BEQ  GOOD
9BDC: C9 DD      189        CMP  #"]"
9BDE: D0 26      190        BNE  FIXEXIT          ; PROMPT DOESN'T MATCH, EXIT
9BE0: A2 00      191        GOOD  LDX  #$00
9BE2: 8E 29 9C   192        STX  QTEFLG          ; CLEAR QUOTE FLAG
9BE5: BD 00 02   193        FIXLOOP LDA  IN,X          ; GET CHARACTER
9BE8: C9 A2      194        CMP  #$A2          ; IF QUOTE THEN FLIP QUOTE FLAG
9BEA: D0 06      195        BNE  NTQTE
9BEC: 4D 29 9C   196        EOR  QTEFLG
9BEF: 8D 29 9C   197        STA  QTEFLG
9BF2: 2C 29 9C   198        NTQTE  BIT  QTEFLG          ; CHECK FOR CONVERSION
9BF5: 30 0C      199        BMI  NEXTIN          ; NO, SKIP TO NEXT
9BF7: BD 00 02   200        LDA  IN,X          ; CONVERT TO UPPER CASE
9BFA: C9 E0      201        CMP  #$E0
9BFC: 90 05      202        BLT  NEXTIN
9BFE: 29 DF      203        AND  #$DF
9C00: 9D 00 02   204        STA  IN,X
9C03: E8          205        NEXTIN INX                ; CONTINUE TO END OF BUFFER
9C04: D0 DF      206        BNE  FIXLOOP
9C06: 68          207        FIXEXIT PLA
9C07: AE 24 9C   208        LDX  XSAVE
9COA: 60          209        RTS
          210 *
          211 *****
          212 *
          213 *          OUTPUT ENTRY POINT
          214 *
          215 *****
          216 *
9COB: 8C 25 9C   217        COUT2  STY  YSAVE
9COE: AC 27 9C   218        LDY  FLAGS          ; CHECK GETLN FLAG
9C11: 10 08      219        BPL  DONE          ; IF CLEAR THEN SKIP
9C13: AC 26 9C   220        LDY  OLDCHAR          ; GET LAST CHARACTER FROM GETLN
9C16: C0 E0      221        CPY  #$E0          ; IF IT IS LOWER CASE THEN USE IT
9C18: 90 01      222        BLT  DONE
9C1A: 98          223        TYA
9C1B: AC 25 9C   224        DONE  LDY  YSAVE
9C1E: 4E 27 9C   225        LSR  FLAGS
9C21: 4C FO FD   226        JMP  COUT1          ; OUTPUT CHARACTER
9C24: 00          227        XSAVE  HEX  00
9C25: 00          228        YSAVE  HEX  00
9C26: 00          229        OLDCHAR HEX  00
9C27: 00          230        FLAGS  HEX  00
9C28: 00          231        ESCFLG HEX  00
9C29: 00          232        QTEFLG HEX  00
          233        LEN    EQU  *-STARTUP
          234        END    EQU  LEN+START

```

--END ASSEMBLY--

ERRORS: 0

335 BYTES

SYMBOL TABLE - ALPHABETICAL ORDER:

|         |         |         |         |         |         |            |         |
|---------|---------|---------|---------|---------|---------|------------|---------|
| A1H     | =\$3D   | A1L     | =\$3C   | A2H     | =\$3F   | A2L        | =\$3E   |
| A4H     | =\$43   | A4L     | =\$42   | AMPER   | =\$03F5 | BASL       | =\$28   |
| COUT1   | =\$FDF0 | COUT2   | =\$9C0B | CSWL    | =\$36   | DEST       | =\$9B00 |
| DONE    | =\$9C1B | END     | =\$614F | ESC     | =\$9BA7 | ESC1       | =\$9BB2 |
| ESC2    | =\$9BC8 | ESCFLG  | =\$9C28 | FIXBUFF | =\$9BD5 | FIXEXIT    | =\$9C06 |
| FIXLOOP | =\$9BE5 | FLAGS   | =\$9C27 | GETLN   | =\$9B76 | GOOD       | =\$9BE0 |
| IN      | =\$0200 | INITIAL | =\$9B2A | KEYIN   | =\$FD1B | KEYIN1     | =\$9B4C |
| KSWL    | =\$38   | LEN     | =\$012A | LOOP    | =\$9BBD | MOVE       | =\$FE2C |
| NEXTIN  | =\$9C03 | NOTPICK | =\$9B8C | NTGETLN | =\$9B98 | NTQTE      | =\$9BF2 |
| OLDCHAR | =\$9C26 | PROMPT  | =\$33   | QTEFLG  | =\$9C29 | ? RELOCATE | =\$6000 |
| RESTART | =\$9B24 | RSVECH  | =\$03F3 | RSVECL  | =\$03F2 | SKIP       | =\$9BBB |
| SKIP1   | =\$9B58 | START   | =\$6025 | STARTUP | =\$9B00 | XSAVE      | =\$9C24 |
| XTBL    | =\$9BD1 | YSAVE   | =\$9C25 |         |         |            |         |

SYMBOL TABLE - NUMERICAL ORDER:

|         |         |         |         |         |         |            |         |
|---------|---------|---------|---------|---------|---------|------------|---------|
| BASL    | =\$28   | PROMPT  | =\$33   | CSWL    | =\$36   | KSWL       | =\$38   |
| A1L     | =\$3C   | A1H     | =\$3D   | A2L     | =\$3E   | A2H        | =\$3F   |
| A4L     | =\$42   | A4H     | =\$43   | LEN     | =\$012A | IN         | =\$0200 |
| RSVECL  | =\$03F2 | RSVECH  | =\$03F3 | AMPER   | =\$03F5 | ? RELOCATE | =\$6000 |
| START   | =\$6025 | END     | =\$614F | DEST    | =\$9B00 | STARTUP    | =\$9B00 |
| RESTART | =\$9B24 | INITIAL | =\$9B2A | KEYIN1  | =\$9B4C | SKIP1      | =\$9B58 |
| GETLN   | =\$9B76 | NOTPICK | =\$9B8C | NTGETLN | =\$9B98 | ESC        | =\$9BA7 |
| ESC1    | =\$9BB2 | SKIP    | =\$9BBB | LOOP    | =\$9BBD | ESC2       | =\$9BC8 |
| XTBL    | =\$9BD1 | FIXBUFF | =\$9BD5 | GOOD    | =\$9BE0 | FIXLOOP    | =\$9BE5 |
| NTQTE   | =\$9BF2 | NEXTIN  | =\$9C03 | FIXEXIT | =\$9C06 | COUT2      | =\$9C0B |
| DONE    | =\$9C1B | XSAVE   | =\$9C24 | YSAVE   | =\$9C25 | OLDCHAR    | =\$9C26 |
| FLAGS   | =\$9C27 | ESCFLG  | =\$9C28 | QTEFLG  | =\$9C29 | KEYIN      | =\$FD1B |
| COUT1   | =\$FDF0 | MOVE    | =\$FE2C |         |         |            |         |

## Appendix X: Supporting Software

### Section X.c Apple Writer Modify

The Apple Writer Modify program contains patches for Apple Writer which will permit lower case entry and display. You should type the following:

BRUN APPLE WRITER MODIFY

Just follow the instructions it gives you. You will be requested to swap discs several times. At the end of this process, a copy of Apple Writer will appear on the Enhancer ][ Utilities Disc. Your original Apple Writer disc is not altered.

```

6      *****
7      *
8      *      APPLE WRITER MODIFIER      *
9      *
10     *      10/30/81   19:00      *
11     *
12     *****
13     *
14     BASEL      EQU   $00
15     BASEH      EQU   $01
16     HOME       EQU   $FC58
17     RDKEY      EQU   $FD0C
18     KEYIN      EQU   $FD1B
19     COUT       EQU   $FDED
20     COUT1      EQU   $FDF0
21     *
22             ORG   $6000
23             OBJ   $6000
24     *
6000: 20 58 FC 25      JSR   HOME      ; CLEAR SCREEN
6003: 20 94 60 26      JSR   HEADER    ; PRINT PAGE HEADER
6006: 20 9B 60 27      JSR   AWMMSG    ; PRINT INSERT APPLE WRITER
6009: 20 AF 60 28      JSR   BLD      ; BLOAD TEDITOR
600C: 20 BD 60 29      JSR   TED
600F: 20 8F 60 30      JSR   CROUT
6012: A2 15 31        LDX   #$15      ; MAKE CHANGES
6014: A0 00 32        LDY   #$00
6016: BD 20 62 33      TMLLOOP LDA   TLADR,X      ; GET ADDRESS TO CHANGE
6019: 85 00 34        STA   BASEL
601B: BD 36 62 35      LDA   THADR,X
601E: 85 01 36        STA   BASEH
6020: BD 4C 62 37      LDA   TPATCH,X      ; GET CHANGE
6023: 91 00 38        STA   (BASEL),Y      ; MAKE CHANGE
6025: CA 39          DEX
6026: 10 EE 40        BPL   TMLLOOP      ; CONTINUE
6028: 20 6C 60 41      JSR   GMOD      ; ADD GENERAL MODIFICATION
602B: 20 B6 60 42      JSR   BSV      ; BSAVE TEDITOR
602E: 20 BD 60 43      JSR   TED
6031: 20 CB 60 44      JSR   SFX
6034: 20 9B 60 46      *      JSR   AWMMSG      ; PRINT INSERT APPLE WRITER
6037: 20 AF 60 47      JSR   BLD      ; BLOAD PRINTER
603A: 20 C4 60 48      JSR   PRNT
603D: 20 8F 60 49      JSR   CROUT
6040: A2 07 50        LDX   #$07
6042: A0 00 51        LDY   #$00
6044: BD A0 62 52      PMLLOOP LDA   PLADR,X      ; GET ADDRESS TO CHANGE
6047: 85 00 53        STA   BASEL
6049: BD A8 62 54      LDA   PHADR,X
604C: 85 01 55        STA   BASEH
604E: BD B0 62 56      LDA   PPATCH,X      ; GET CHANGE
6051: 91 00 57        STA   (BASEL),Y      ; MAKE CHANGE
6053: CA 58          DEX
6054: 10 EE 59        BPL   PMLLOOP      ; CONTINUE
6056: 20 6C 60 60      JSR   GMOD      ; ADD GENERAL MODIFICATION
6059: 20 B6 60 61      JSR   BSV      ; BSAVE PRINTER
605C: 20 C4 60 62      JSR   PRNT
605F: 20 CB 60 63      JSR   SFX

```

|       |    |    |     |         |          |                                      |
|-------|----|----|-----|---------|----------|--------------------------------------|
| 6062: | A9 | 60 | 64  | LDA     | #>ENDMSG |                                      |
| 6064: | A0 | E0 | 65  | LDY     | #<ENDMSG |                                      |
| 6066: | 20 | 7A | 60  | JSR     | MESSAGE  | ; PRINT END MESSAGE                  |
| 6069: | 4C | D0 | 03  | JMP     | \$3D0    | ; JUMP TO BASIC                      |
|       |    |    | 68  | *       |          |                                      |
| 606C: | A0 | 4F | 69  | GMOD    | LDY      | #END-OUTPATCH ; GENERAL MODIFICATION |
| 606E: | B9 | C8 | 61  | MDLOOP  | LDA      | GPATCH,Y ; GET BYTE                  |
| 6071: | 99 | 20 | 18  | STA     | \$1820,Y | ; STORE IT IN TEDITOR OR PRINTER     |
| 6074: | 88 |    | 72  | DEY     |          |                                      |
| 6075: | 10 | F7 | 73  | BPL     | MDLOOP   | ; CONTINUE UNTIL DONE                |
| 6077: | 4C | A5 | 60  | JMP     | EUMSG    | ; INSERT ENHANCER UTILITY            |
|       |    |    | 75  | *       |          |                                      |
| 607A: | 8D | 84 | 60  | MESSAGE | STA      | MSGLOOP+2                            |
| 607D: | 8C | 83 | 60  |         | STY      | MSGLOOP+1                            |
| 6080: | A0 | 00 | 78  |         | LDY      | #\$00                                |
| 6082: | B9 | 29 | 61  | MSGLOOP | LDA      | MSG1,Y                               |
| 6085: | F0 | 07 | 80  |         | BEQ      | RTS1                                 |
| 6087: | 20 | ED | FD  |         | JSR      | COUT                                 |
| 608A: | C8 |    | 82  |         | INY      |                                      |
| 608B: | 4C | 82 | 60  |         | JMP      | MSGLOOP                              |
|       |    |    | 84  | *       |          |                                      |
| 608E: | 60 |    | 85  | RTS1    | RTS      |                                      |
|       |    |    | 86  | *       |          |                                      |
| 608F: | A9 | 8D | 87  | CROUT   | LDA      | #\$8D                                |
| 6091: | 4C | ED | FD  |         | JMP      | COUT                                 |
|       |    |    | 89  | *       |          |                                      |
| 6094: | A9 | 61 | 90  | HEADER  | LDA      | #>HEAD                               |
| 6096: | A0 | 0B | 91  |         | LDY      | #<HEAD                               |
| 6098: | 4C | 7A | 60  |         | JMP      | MESSAGE                              |
|       |    |    | 93  | *       |          |                                      |
| 609B: | A9 | 61 | 94  | AWMSG   | LDA      | #>MSG1                               |
| 609D: | A0 | 29 | 95  |         | LDY      | #<MSG1                               |
| 609F: | 20 | 7A | 60  |         | JSR      | MESSAGE                              |
| 60A2: | 4C | D2 | 60  |         | JMP      | KEYWAIT                              |
|       |    |    | 98  | *       |          |                                      |
| 60A5: | A9 | 61 | 99  | EUMSG   | LDA      | #>MSG2                               |
| 60A7: | A0 | 5C | 100 |         | LDY      | #<MSG2                               |
| 60A9: | 20 | 7A | 60  |         | JSR      | MESSAGE                              |
| 60AC: | 4C | D2 | 60  |         | JMP      | KEYWAIT                              |
|       |    |    | 103 | *       |          |                                      |
| 60AF: | A9 | 61 | 104 | BLD     | LDA      | #>BLOAD                              |
| 60B1: | A0 | 93 | 105 |         | LDY      | #<BLOAD                              |
| 60B3: | 4C | 7A | 60  |         | JMP      | MESSAGE                              |
|       |    |    | 107 | *       |          |                                      |
| 60B6: | A9 | 61 | 108 | BSV     | LDA      | #>BSAVE                              |
| 60B8: | A0 | 9B | 109 |         | LDY      | #<BSAVE                              |
| 60BA: | 4C | 7A | 60  |         | JMP      | MESSAGE                              |
|       |    |    | 111 | *       |          |                                      |
| 60BD: | A9 | 61 | 112 | TED     | LDA      | #>TEDIT                              |
| 60BF: | A0 | A3 | 113 |         | LDY      | #<TEDIT                              |
| 60C1: | 4C | 7A | 60  |         | JMP      | MESSAGE                              |
|       |    |    | 115 | *       |          |                                      |
| 60C4: | A9 | 61 | 116 | PRNT    | LDA      | #>PRINT                              |
| 60C6: | A0 | B1 | 117 |         | LDY      | #<PRINT                              |
| 60C8: | 4C | 7A | 60  |         | JMP      | MESSAGE                              |
|       |    |    | 119 | *       |          |                                      |
| 60CB: | A9 | 61 | 120 | SFX     | LDA      | #>SUFFIX                             |
| 60CD: | A0 | BF | 121 |         | LDY      | #<SUFFIX                             |

|                |     |         |     |                                |
|----------------|-----|---------|-----|--------------------------------|
| 60CF: 4C 7A 60 | 122 |         | JMP | MESSAGE                        |
|                | 123 | *       |     |                                |
| 60D2: 2C 10 C0 | 124 | KEYWAIT | BIT | \$C010                         |
| 60D5: A4 24    | 125 |         | LDY | \$24                           |
| 60D7: A9 60    | 126 |         | LDA | #\$60                          |
| 60D9: 91 28    | 127 |         | STA | (\$28),Y                       |
| 60DB: A9 A0    | 128 |         | LDA | #" "                           |
| 60DD: 4C 1B FD | 129 |         | JMP | KEYIN                          |
|                | 130 | *       |     |                                |
| 60E0: 8D 8D    | 131 | ENDMSG  | HEX | 8D8D                           |
| 60E2: C1 F0 F0 |     |         |     |                                |
| 60E5: EC E5 A0 |     |         |     |                                |
| 60E8: D7 F2 E9 |     |         |     |                                |
| 60EB: F4 E5 F2 |     |         |     |                                |
| 60EE: A0 ED EF |     |         |     |                                |
| 60F1: E4 E9 E6 |     |         |     |                                |
| 60F4: E9 E3 E1 |     |         |     |                                |
| 60F7: F4 E9 EF |     |         |     |                                |
| 60FA: EE F3    | 132 |         | ASC | "Apple Writer modifications"   |
| 60FC: 8D       | 133 |         | HEX | 8D                             |
| 60FD: E1 F2 E5 |     |         |     |                                |
| 6100: A0 E3 EF |     |         |     |                                |
| 6103: ED F0 EC |     |         |     |                                |
| 6106: E5 F4 E5 | 134 |         | ASC | "are complete"                 |
| 6109: 8D 00    | 135 |         | HEX | 8D00                           |
|                | 136 | *       |     |                                |
| 610B: C1 F0 F0 |     |         |     |                                |
| 610E: EC E5 A0 |     |         |     |                                |
| 6111: D7 F2 E9 |     |         |     |                                |
| 6114: F4 E5 F2 |     |         |     |                                |
| 6117: A0 CD EF |     |         |     |                                |
| 611A: E4 E9 E6 |     |         |     |                                |
| 611D: F9 A0 D0 |     |         |     |                                |
| 6120: F2 EF E7 |     |         |     |                                |
| 6123: F2 E1 ED | 137 | HEAD    | ASC | "Apple Writer Modify Program"  |
| 6126: 8D 8D 00 | 138 |         | HEX | 8D8D00                         |
| 6129: 8D 8D    | 139 | MSG1    | HEX | 8D8D                           |
| 612B: C9 EE F3 |     |         |     |                                |
| 612E: E5 F2 F4 |     |         |     |                                |
| 6131: A0 C1 F0 |     |         |     |                                |
| 6134: F0 EC E5 |     |         |     |                                |
| 6137: A0 D7 F2 |     |         |     |                                |
| 613A: E9 F4 E5 |     |         |     |                                |
| 613D: F2 A0 E4 |     |         |     |                                |
| 6140: E9 F3 EB |     |         |     |                                |
| 6143: E5 F4 F4 |     |         |     |                                |
| 6146: E5       | 140 |         | ASC | "Insert Apple Writer diskette" |
| 6147: 8D       | 141 |         | HEX | 8D                             |
| 6148: E1 EE E4 |     |         |     |                                |
| 614B: A0 F0 F2 |     |         |     |                                |
| 614E: E5 F3 F3 |     |         |     |                                |
| 6151: A0 DB D2 |     |         |     |                                |
| 6154: C5 D4 D5 |     |         |     |                                |
| 6157: D2 CE DD |     |         |     |                                |
| 615A: A0       | 142 |         | ASC | "and press [RETURN] "          |
| 615B: 00       | 143 |         | HEX | 00                             |
|                | 144 | *       |     |                                |
| 615C: 8D 8D    | 145 | MSG2    | HEX | 8D8D                           |

|       |          |     |                              |                           |
|-------|----------|-----|------------------------------|---------------------------|
| 615E: | C9 EE F3 |     |                              |                           |
| 6161: | E5 F2 F4 |     |                              |                           |
| 6164: | A0 C5 EE |     |                              |                           |
| 6167: | E8 E1 EE |     |                              |                           |
| 616A: | E3 E5 F2 |     |                              |                           |
| 616D: | A0 D5 F4 |     |                              |                           |
| 6170: | E9 EC E9 |     |                              |                           |
| 6173: | F4 F9    | 146 | ASC                          | "Insert Enhancer Utility" |
| 6175: | A0 E4 E9 |     |                              |                           |
| 6178: | F3 EB E5 |     |                              |                           |
| 617B: | F4 F4 E5 | 147 | ASC                          | " diskette"               |
| 617E: | 8D       | 148 | HEX                          | 8D                        |
| 617F: | E1 EE E4 |     |                              |                           |
| 6182: | A0 F0 F2 |     |                              |                           |
| 6185: | E5 F3 F3 |     |                              |                           |
| 6188: | A0 DB D2 |     |                              |                           |
| 618B: | C5 D4 D5 |     |                              |                           |
| 618E: | D2 CE DD |     |                              |                           |
| 6191: | A0       | 149 | ASC                          | "and press [RETURN] "     |
| 6192: | 00       | 150 | HEX                          | 00                        |
|       |          | 151 | *                            |                           |
| 6193: | 8D 84    | 152 | BLOAD                        | HEX 8D84                  |
| 6195: | C2 CC CF |     |                              |                           |
| 6198: | C1 C4    | 153 | ASC                          | "BLOAD"                   |
| 619A: | 00       | 154 | HEX                          | 00                        |
|       |          | 155 | *                            |                           |
| 619B: | 8D 84    | 156 | BSAVE                        | HEX 8D84                  |
| 619D: | C2 D3 C1 |     |                              |                           |
| 61A0: | D6 C5    | 157 | ASC                          | "BSAVE"                   |
| 61A2: | 00       | 158 | HEX                          | 00                        |
|       |          | 159 | *                            |                           |
| 61A3: | D4 C5 C4 |     |                              |                           |
| 61A6: | C9 D4 CF |     |                              |                           |
| 61A9: | D2 AC C1 |     |                              |                           |
| 61AC: | A4 B8 B0 |     |                              |                           |
| 61AF: | B3       | 160 | TEDIT                        | ASC "TEDITOR,A\$803"      |
| 61B0: | 00       | 161 | HEX                          | 00                        |
|       |          | 162 | *                            |                           |
| 61B1: | D0 D2 C9 |     |                              |                           |
| 61B4: | CE D4 C5 |     |                              |                           |
| 61B7: | D2 AC C1 |     |                              |                           |
| 61BA: | A4 B8 B0 |     |                              |                           |
| 61BD: | B3       | 163 | PRINT                        | ASC "PRINTER,A\$803"      |
| 61BE: | 00       | 164 | HEX                          | 00                        |
|       |          | 165 | *                            |                           |
| 61BF: | AC CC A4 |     |                              |                           |
| 61C2: | B1 B0 B7 |     |                              |                           |
| 61C5: | B0       | 166 | SUFFIX                       | ASC ",L\$1070"            |
| 61C6: | 8D       | 167 | HEX                          | 8D                        |
| 61C7: | 00       | 168 | HEX                          | 00                        |
|       |          | 169 | *                            |                           |
|       |          | 170 | *****                        |                           |
|       |          | 171 | *                            | *                         |
|       |          | 172 | * Apple Writer modifications | *                         |
|       |          | 173 | *                            | *                         |
|       |          | 174 | *                            | *                         |
|       |          | 175 | *****                        |                           |
|       |          | 176 | *                            |                           |

|                |     |          |     |          |
|----------------|-----|----------|-----|----------|
|                | 177 | *        |     |          |
|                | 178 | GPATCH   | ORG | \$1820   |
|                | 179 | *        |     |          |
| 1820: C9 E0    | 180 | OUTPATCH | CMP | #\$E0    |
| 1822: 90 02    | 181 |          | BCC | SKIP1    |
| 1824: 29 BF    | 182 |          | AND | #\$BF    |
| 1826: C9 C0    | 183 | SKIP1    | CMP | #\$C0    |
| 1828: 90 02    | 184 |          | BCC | SKIP2    |
| 182A: 09 20    | 185 |          | ORA | #\$20    |
| 182C: C9 40    | 186 | SKIP2    | CMP | #\$40    |
| 182E: B0 04    | 187 |          | BCS | SKIP3    |
| 1830: 49 20    | 188 |          | EOR | #\$20    |
| 1832: 69 A0    | 189 |          | ADC | #\$A0    |
| 1834: 60       | 190 | SKIP3    | RTS |          |
|                | 191 | *        |     |          |
| 1835: C9 E0    | 192 | INPATCH  | CMP | #\$E0    |
| 1837: 90 03    | 193 |          | BCC | SKIP4    |
| 1839: 29 DF    | 194 |          | AND | #\$DF    |
| 183B: 60       | 195 |          | RTS |          |
|                | 196 | *        |     |          |
| 183C: C9 C0    | 197 | SKIP4    | CMP | #\$C0    |
| 183E: 90 03    | 198 |          | BCC | SKIP5    |
| 1840: 29 1F    | 199 |          | AND | #\$1F    |
| 1842: 60       | 200 |          | RTS |          |
|                | 201 | *        |     |          |
| 1843: C9 A0    | 202 | SKIP5    | CMP | #\$A0    |
| 1845: 90 ED    | 203 |          | BCC | SKIP3    |
| 1847: 09 40    | 204 |          | ORA | #\$40    |
| 1849: 60       | 205 |          | RTS |          |
|                | 206 | *        |     |          |
| 184A: 20 20 18 | 207 | OUT1     | JSR | OUTPATCH |
| 184D: 4C F0 FD | 208 |          | JMP | COUT1    |
|                | 209 | *        |     |          |
| 1850: 20 20 18 | 210 | OUT2     | JSR | OUTPATCH |
| 1853: 91 28    | 211 |          | STA | (\$28),Y |
| 1855: C8       | 212 |          | INY |          |
| 1856: 60       | 213 |          | RTS |          |
|                | 214 | *        |     |          |
| 1857: A5 10    | 215 | CAPSET   | LDA | \$10     |
| 1859: C9 E0    | 216 |          | CMP | #\$E0    |
| 185B: 90 02    | 217 |          | BCC | SKIP6    |
| 185D: 29 DF    | 218 |          | AND | #\$DF    |
| 185F: C9 C9    | 219 | SKIP6    | CMP | #\$C9    |
| 1861: 60       | 220 |          | RTS |          |
|                | 221 | *        |     |          |
|                | 222 | *        |     |          |
| 1862: 48       | 223 | SHPATCH  | PHA |          |
| 1863: A5 0B    | 224 |          | LDA | \$0B     |
| 1865: D0 04    | 225 |          | BNE | SKIP7    |
| 1867: 68       | 226 |          | PLA |          |
| 1868: 29 3F    | 227 |          | AND | #\$3F    |
| 186A: 60       | 228 |          | RTS |          |
|                | 229 | *        |     |          |
| 186B: 68       | 230 | SKIP7    | PLA |          |
| 186C: 4C 35 18 | 231 |          | JMP | INPATCH  |
|                | 232 | *        |     |          |
| 186F: 00       | 233 | END      | HEX | 00       |
|                | 234 | *        |     |          |



|                |     |        |                       |
|----------------|-----|--------|-----------------------|
|                | 235 | ORG    | \$6220                |
|                | 236 | OBJ    | \$6220                |
|                | 237 | *      |                       |
| 6220: E0       | 238 | TLADR  | HEX E0                |
| 6221: E6 E7 E8 |     |        |                       |
| 6224: E9       | 239 | HEX    | E6E7E8E9              |
| 6225: E6 E7 E8 | 240 | HEX    | E6E7E8                |
| 6228: 05 06 07 | 241 | HEX    | 050607                |
| 622B: 32 33 34 |     |        |                       |
| 622E: 35 36 37 |     |        |                       |
| 6231: 38 39    | 242 | HEX    | 3233343536373839      |
| 6233: 49 4A 4B | 243 | HEX    | 494A4B                |
|                | 244 | *      |                       |
| 6236: 09       | 245 | THADR  | HEX 09                |
| 6237: 09 09 09 |     |        |                       |
| 623A: 09       | 246 | HEX    | 09090909              |
| 623B: 0A 0A 0A | 247 | HEX    | 0A0A0A                |
| 623E: 15 15 15 | 248 | HEX    | 151515                |
| 6241: 15 15 15 |     |        |                       |
| 6244: 15 15 15 |     |        |                       |
| 6247: 15 15    | 249 | HEX    | 1515151515151515      |
| 6249: 15 15 15 | 250 | HEX    | 151515                |
|                | 251 | *      |                       |
|                | 252 | *****  |                       |
|                | 253 | *      | *                     |
|                | 254 | *      | PATCHES FOR TEDITOR * |
|                | 255 | *      | *                     |
|                | 256 | *****  |                       |
|                | 257 | *      |                       |
|                | 258 | TPATCH | ORG \$9E0             |
| 09E0: EA       | 259 |        | NOP                   |
|                | 260 | *      |                       |
|                | 261 |        | ORG \$9E6             |
| 09E6: 20 57 18 | 262 | JSR    | CAPSET                |
| 09E9: EA       | 263 |        | NOP                   |
|                | 264 | *      |                       |
|                | 265 |        | ORG \$AE6             |
| 0AE6: 20 50 18 | 266 | JSR    | OUT2                  |
|                | 267 | *      |                       |
|                | 268 |        | ORG \$1505            |
| 1505: 4C 62 18 | 269 | JMP    | SHPATCH               |
|                | 270 | *      |                       |
|                | 271 |        | ORG \$1532            |
| 1532: EA       | 272 |        | NOP                   |
| 1533: EA       | 273 |        | NOP                   |
| 1534: D0 DF    | 274 | BNE    | \$1515                |
| 1536: 20 01 15 | 275 | JSR    | \$1501                |
| 1539: 48       | 276 |        | PHA                   |
|                | 277 | *      |                       |
|                | 278 |        | ORG \$1549            |
| 1549: 20 4A 18 | 279 | JSR    | OUT1                  |
|                | 280 | *      |                       |
|                | 281 |        | ORG \$62A0            |
|                | 282 |        | OBJ \$62A0            |
|                | 283 | *      |                       |
| 62A0: 9D 9E 9F | 284 | PLADR  | HEX 9D9E9F            |
| 62A3: CA CB    | 285 |        | HEX CACB              |
| 62A5: DD DE DF | 286 |        | HEX DDDDEF            |

```

287 *
62A8: 10 10 10 288 PHADR    HEX  101010
62AB: 10 10    289          HEX  1010
62AD: 10 10 10 290          HEX  101010
291 *
292 *****
293 *                               *
294 *      PATCHES FOR PRINTER    *
295 *                               *
296 *****
297 *
298 PPATCH    ORG  $109D
109D: 4C 35 18 299          JMP  INPATCH
300 *
301          ORG  $10CA
10CA: EA     302          NOP
10CB: EA     303          NOP
304 *
305          ORG  $10DD
10DD: 20 4A 18 306          JSR  OUT1

```

--END ASSEMBLY--

ERRORS: 0

24 BYTES

#### SYMBOL TABLE - ALPHABETICAL ORDER:

|         |         |          |         |         |         |         |         |
|---------|---------|----------|---------|---------|---------|---------|---------|
| AWMSG   | =\$609B | BASEH    | =\$01   | BASEL   | =\$00   | BLD     | =\$60AF |
| BLOAD   | =\$6193 | BSAVE    | =\$619B | BSV     | =\$60B6 | CAPSET  | =\$1857 |
| COUT    | =\$FD1B | COUT1    | =\$FDF0 | CROUT   | =\$608F | END     | =\$186F |
| ENDMSG  | =\$60E0 | EUMSG    | =\$60A5 | GMOD    | =\$606C | GPATCH  | =\$61C8 |
| HEAD    | =\$610B | HEADER   | =\$6094 | HOME    | =\$FC58 | INPATCH | =\$1835 |
| KEYIN   | =\$FD1B | KEYWAIT  | =\$60D2 | MDLOOP  | =\$606E | MESSAGE | =\$607A |
| MSG1    | =\$6129 | MSG2     | =\$615C | MSGLOOP | =\$6082 | OUT1    | =\$184A |
| OUT2    | =\$1850 | OUTPATCH | =\$1820 | PHADR   | =\$62A8 | PLADR   | =\$62A0 |
| PMLLOOP | =\$6044 | PPATCH   | =\$62B0 | PRINT   | =\$61B1 | PRNT    | =\$60C4 |
| ? RDKEY | =\$FD0C | RTS1     | =\$608E | SFX     | =\$60CB | SHPATCH | =\$1862 |
| SKIP1   | =\$1826 | SKIP2    | =\$182C | SKIP3   | =\$1834 | SKIP4   | =\$183C |
| SKIP5   | =\$1843 | SKIP6    | =\$185F | SKIP7   | =\$186B | SUFFIX  | =\$61BF |
| TED     | =\$60BD | TEDIT    | =\$61A3 | THADR   | =\$6236 | TLADR   | =\$6220 |
| TMLOOP  | =\$6016 | TPATCH   | =\$624C |         |         |         |         |

#### SYMBOL TABLE - NUMERICAL ORDER:

|         |         |         |         |          |         |        |         |
|---------|---------|---------|---------|----------|---------|--------|---------|
| BASEL   | =\$00   | BASEH   | =\$01   | OUTPATCH | =\$1820 | SKIP1  | =\$1826 |
| SKIP2   | =\$182C | SKIP3   | =\$1834 | INPATCH  | =\$1835 | SKIP4  | =\$183C |
| SKIP5   | =\$1843 | OUT1    | =\$184A | OUT2     | =\$1850 | CAPSET | =\$1857 |
| SKIP6   | =\$185F | SHPATCH | =\$1862 | SKIP7    | =\$186B | END    | =\$186F |
| TMLOOP  | =\$6016 | PMLLOOP | =\$6044 | GMOD     | =\$606C | MDLOOP | =\$606E |
| MESSAGE | =\$607A | MSGLOOP | =\$6082 | RTS1     | =\$608E | CROUT  | =\$608F |
| HEADER  | =\$6094 | AWMSG   | =\$609B | EUMSG    | =\$60A5 | BLD    | =\$60AF |
| BSV     | =\$60B6 | TED     | =\$60BD | PRNT     | =\$60C4 | SFX    | =\$60CB |
| KEYWAIT | =\$60D2 | ENDMSG  | =\$60E0 | HEAD     | =\$610B | MSG1   | =\$6129 |

|        |         |        |         |         |         |       |         |
|--------|---------|--------|---------|---------|---------|-------|---------|
| MSG2   | =\$615C | BLOAD  | =\$6193 | BSAVE   | =\$619B | TEDIT | =\$61A3 |
| PRINT  | =\$61B1 | SUFFIX | =\$61BF | GPATCH  | =\$61C8 | TLADR | =\$6220 |
| THADR  | =\$6236 | TPATCH | =\$624C | PLADR   | =\$62A0 | PHADR | =\$62A8 |
| PPATCH | =\$62B0 | HOME   | =\$FC58 | ? RDKEY | =\$FDOC | KEYIN | =\$FD1B |
| COUT   | =\$FDED | COUT1  | =\$FDF0 |         |         |       |         |

## Appendix X: Supporting Software

### Section X.d The Macro editor

Included on the Enhancer ][ utility diskette is an Applesoft program to create, add, or modify predefined keystrokes for use as Macro keys. To use the program just type, RUN MACRO EDITOR and wait for the main menu to be displayed on the screen. It is self explanatory and does not need a detailed explanation. Just depress the letter corresponding to what function you want. There is no need to press the return key. Pressing the ESCAPE key will immediatly return you to the main menu from most of the program.

#### Section X.d.1 The Edit Mode

Depressing the E key from the main menu will put you in the edit mode. The editor will display at the top of the screen the amount of memory which you have used and the amount of memory that remains free for definitions. Editor mode definitions are explained below:

**LOWERCASE:** depress the L key to activate/deactivate this mode. When this option is displayed in inverse video, you are accessing macros in the Lower Case Mode.

**CONTROL:** depress the C key to activate/deactivate this mode. When this option is displayed in inverse video, the keyboard character for which you are about to define a macro will be one that has the control key pressed.

**SHIFT:** depress the S key to activate/deactivate this mode. When this option is displayed in inverse video, the keyboard character for which you are about to define a macro will be one that has the shift key pressed.

**KEYPAD:** depress the K key to activate/deactivate this mode. When this option is displayed in inverse video, the keyboard character for which your are about to define a macro must come from an optional keypad input port on the Enhancer ][ (keypad not available from Videx). You may, therefore, only access the macro key from an optional keypad that is connected to the Enhancer ][.

**Note:** Keypad Mode automatically deselects all the other three modes. Likewise, any of the other modes will deselect Keypad Mode.

When you have selected the modes above that you want to use, depress the return key. The editor will ask which character key you want to use.

## Appendix X: Supporting Software

Example: Suppose we want to define a control D in the Lower Case Mode. Here are our keystroke sequences (starting from the main menu):

```
E
C
L
<cr>
D
```

The E selects Edit Mode. The C selects the Control Mode (i.e. CONTROL is displayed in inverse). The L selects Lower Case Mode (not for the keyboard, however). The <cr> ends the Edit Menu selection. The D selects the keyboard character key.

Once you have selected the Edit Mode and character key, the editor will search through memory to see if that key has already been defined. If it has, it will be displayed with the cursor to the right of the definition, otherwise, the cursor will appear along the left edge of the screen. The macro is now open for editing. To type in a definition just type out the key sequence normally - well, almost normally.

For example, if you had defined a shift-C for your macro access key, you might type in the word CATALOG <cr> for your macro. If you do this, you will notice that the RETURN key gets printed out as an inverse M on the screen. This is because the RETURN is a control-M. The editor prints out all control characters as inverse characters. Any control character may be directly entered from the keyboard except for the left and right arrow keys (i.e. ^H and ^U), ^X, ^C, ^O, and ESCAPE (i.e. ^[]). Below is a brief explanation of each key and what they do.

CONTROL-C (^C): Accept. Once you have finished defining your macro, type ^C to accept the definition and return you to the main menu.

ESCAPE (^[]): Depressing this key will delete the macro and return you to the main menu.

CONTROL-X (^X): Depressing this key will delete the macro definition and return the cursor to the beginning of the macro define sequence, with the mode and macro access key already selected.

LEFT AND RIGHT ARROW (^H and ^U): These keys allow you to move through the text of the macro. With the Left arrow deleting characters and the Right arrow key restoring them.

## Appendix X: Supporting Software

CONTROL-0 (^0): Override. Once you type in a ^0, the editor will accept any character you wish - including the command keys described above - but you may only enter one at a time. So, if you want to place two ^C's right next to each other, you must press the ^0 before each occurrence of a ^C else the editor will process the ^C as an Accept and return you to the Main Menu.

### Section X.d.2 The Display Mode.

Typing D from the main menu will put you in the display mode. The program will search through memory for the macros you have defined and display them on the screen. Under the mode column, the options that have been selected are indicated by the first letter of the option (i.e. L meaning lowercase) being shown for every option you have selected, if you haven't selected the option, it will not be printed. Following the mode options is the macro access key, or the key that will have to be pressed to activate the macro. And finally, the description of the macro itself with all control characters shown in inverse video. Pressing any key will halt or resume the listing, with the Escape key returning you to the main menu.

### Section X.d.3 The Catalog Mode.

Typing C from the main menu will catalog the current disc drive.

### Section X.d.4 Save Macros to Disc

Typing S from the main menu will save the current macro in memory (not the macros in the Enhancer) to the disc. The main menu will be replaced with a new menu showing download options you may wish to include in your macro file [appendix: C]. To select an option, just depress the corresponding number next to the option and the option selected will be shown in inverse video. Once you have selected the download options you want, you may continue the save operation by pressing the RETURN key. The prompt: "ENTER FILENAME: MACRO." will appear. At this point you may type in whatever filename you want for your Macro and a BRUN able file will be saved to the disc. If you don't want your file prefixed with "MACRO." you may use the Left arrow to backover it or use ^X to remove it from the filename.

### Section X.d.5 Load Macros from Disk

Typing L from the main menu will give you the same "ENTER FILENAME: MACRO." prompt that you get in the Save mode. To load in your macro just type in your filename and press the

## Appendix X: Supporting Software

RETURN key. If you don't need the automatic "MACRO." prefixed to your file name, you may back over it with the left arrow key or use the ^X to remove it. Also, if you want to get a quick catalog of you disc, you may type in a return without typing in a filename. You will be returned to the "ENTER FILENAME:" prompt.

### Section X.d.6 Quit

Typing Q from the main menu will unceremoniously dump you out into Applesoft Basic. If you want to re-enter the program type in GOTO 1000 and you will be back in the main menu. However, if you type GOTO 1000 and the program does not function properly, it is likely that the program variables have been altered or lost, and you will have to type RUN to re-initialize the variables so the program will work properly.

### Section X.d.7 Macro Down Load

Typing M from the main menu will automatically down load the current macro in memory to the Enhancer. This option is only available after a macro file has been saved or loaded from disc.

### Section X.d.8 Program Errata

Since the Macro Editor is written in Applesoft Basic it is possible to tailor the program for your own use. The program is commented and structured in a logical order. The last few REM statements in the program give an abbreviated list of the variables and what they stand for. However, we also want to say that small changes in a program can lead to unexpected results, so please be careful when you make changes. Murphy's laws are not a figment of the imagination. If you want to speed up the Editor you can compile it with Microsoft's TASC without modification using its default values. We estimate an approximate 300% increase in execution speed for many of its functions.

Note: The down load program resides in memory from \$8CA0 TO \$8F90.

# LIST

```

1 HIMEM: 30000: GOTO 25000
2 REM *****
3 REM *
4 REM * ENHANCER ][ MACRO EDITOR *
5 REM *
6 REM * NOVEMBER 4, 1981 *
7 REM *
8 REM *****
10 FOR I = 1 TO 512:AP = AP + 1
20 IF PEEK (AD + AP) < 127 THEN I = 512
30 NEXT I: RETURN
40 IF CHR < 32 THEN INVERSE :CHR = CHR + 64
50 PRINT CHR$ (CHR);: NORMAL : RETURN
100 AP = 2:FA = 0: REM * SEARCH FOR SIMILAR MACRO KEY *
110 IF AP > = 512 - BU THEN FA = 1: RETURN
120 PO = AP
130 IF PEEK (AD + AP) = MK AND PEEK (AD + AP - 1) = MO THEN 170
140 GOSUB 10: REM SEARCH FOR HI BIT CLEAR +1
150 IF AP > = 512 - BU THEN FA = 1: RETURN
160 GOTO 120
170 IF AP > = 512 - BU THEN FA = 1: RETURN
180 RETURN
200 PS = AP - 1:PO = PS + 2: REM * REMOVE MACRO & COMPACT BUFFER *
210 IF PEEK (AD + PO) > 127 THEN PO = PO + 1: GOTO 210
215 IF PO > = 512 - BU THEN BU = 512 - PS: RETURN
220 FOR PI = 0 TO 512 - BU - PO
230 POKE (AD + PS + PI), PEEK (AD + PO + PI): NEXT
240 BU = BU + PO - PS: RETURN
300 IF AL = 1 THEN INVERSE
310 VTAB 5: HTAB 07: PRINT "LOWERCASE": NORMAL
320 IF CN = 1 THEN INVERSE
330 VTAB 5: HTAB 18: PRINT "CONTROL": NORMAL
340 IF SH = 1 THEN INVERSE
350 VTAB 5: HTAB 27: PRINT "SHIFT": NORMAL
360 IF KP = 1 THEN INVERSE
370 VTAB 5: HTAB 34: PRINT "KEYPAD": NORMAL
380 MO = KP * 8 + AL * 4 + SH * 2 + CN
390 RETURN
400 PS = AP + 1:PO = 1
410 CHR = PEEK (AD + PS): IF CHR < 128 THEN 490
420 NA$(PO) = CHR:PS = PS + 1:PO = PO + 1: IF PS > = 512 - BU THEN 490
430 GOTO 410
490 NA$(0) = PO - 1: RETURN
500 PS = 514 - BU: POKE AD + PS - 2,MO: POKE AD + PS - 1,MK
510 FOR P = 1 TO NA$(0): POKE AD + PS + P - 1,NA$(P): NEXT :BU = BU - NA
    $(0) - 2: POKE AD + 512 - BU,0
520 RETURN
700 Q = MO:DU$ = ""
710 FOR P = 1 TO 4
720 IF Q - INT (Q / 2) * 2 THEN DU$ = QA$(P) + DU$
740 Q = INT (Q / 2)
750 NEXT : HTAB 2: PRINT DU$: RETURN
800 MK = - 1: FOR P = 0 TO 15
810 IF KA(P) = KY THEN MK = P
820 NEXT : RETURN
900 POKE - 16368,0: REM * GET BUT DON'T SHOW CURSOR *

```



```

910 IF PEEK ( - 16384) < 128 THEN 910
920 IF PEEK ( - 16384) = 155 THEN POP : GOTO 1000
930 POKE - 16368,0:GC$ = CHR$ ( PEEK ( - 16384)): RETURN
1000 REM

```

\* ENTRY POINT FOR MENU \*

```

1010 TEXT : HOME : POKE 34,2: INVERSE : PRINT : HTAB 9: PRINT "ENHANCER
] [ MACRO EDITOR": POKE - 16368,0: NORMAL : IF LEN (NA$) THEN PRINT
: HTAB (35 - LEN (NA$)) / 2: PRINT "File: "NA$
1020 VTAB 08: HTAB 10: PRINT "E - EDIT MACROS"
1040 VTAB 09: HTAB 10: PRINT "D - DISPLAY MACROS"
1050 VTAB 10: HTAB 10: PRINT "C - CATALOG DISK"
1060 VTAB 11: HTAB 10: PRINT "S - SAVE MACROS TO DISK"
1070 VTAB 12: HTAB 10: PRINT "L - LOAD MACROS FROM DISK"
1080 VTAB 13: HTAB 10: PRINT "Q - QUIT EDITOR"
1085 IF LEN (NA$) THEN VTAB 14: HTAB 10: PRINT "M - MACRO DOWN LOAD"
1090 EN = 1023 - BU:EL = EN - INT (EN / 256) * 256:EH = EN / 256
1095 POKE AD - 1,EL: POKE AD,EH
1100 VTAB 20: HTAB 05: PRINT "SELECT OPTION:": GET GC$: IF ASC (GC$) >
95 THEN GC$ = CHR$ ( ASC (GC$) - 32)
1110 IF GC$ = "C" THEN : PRINT : PRINT D$"CATALOG": GOSUB 19100: GOTO 10
00
1120 IF GC$ = "Q" THEN TEXT : HOME : END
1130 IF GC$ = "L" THEN 13000
1140 IF GC$ = "S" THEN 3000
1150 IF GC$ = "E" THEN 5000
1160 IF GC$ = "D" THEN 4000
1170 IF GC$ = "M" AND LEN (NA$) THEN HOME : PRINT : PRINT "Are you sur
e? ": GOSUB 900: IF GC$ = "Y" OR GC$ = "y" THEN PRINT : PRINT "Dow
n Loading.": CALL 36000
1200 GOTO 1000: REM * MAIN MENU *
3000 REM

```

\* DOWN LOAD OPTIONS \*

```

3010 HOME : HTAB 11: INVERSE : PRINT "DOWN LOAD OPTIONS": NORMAL : POKE
34,3: PRINT : PRINT
3500 Q = ST
3510 FOR P = 1 TO 7
3520 B(P) = Q - INT (Q / 2) * 2:Q = INT (Q / 2)
3530 NEXT
3600 VTAB 5: HTAB 2: PRINT "1 ";: IF B(1) = 1 THEN INVERSE
3610 PRINT "DISABLE SHIFT LOCK": NORMAL
3620 VTAB 7: HTAB 2: PRINT "2 ";: IF B(2) = 1 THEN INVERSE
3630 PRINT "LOCK KEYBOARD MODE": NORMAL
3640 VTAB 9: HTAB 2: PRINT "3 ";: IF B(3) = 1 THEN INVERSE
3650 PRINT "SELECT DEFAULT TO LOWER CASE": NORMAL
3660 VTAB 11: HTAB 2: PRINT "4 ";: IF B(4) = 1 THEN INVERSE
3670 PRINT "DISABLE AUTO REPEAT": NORMAL
3680 VTAB 13: HTAB 2: PRINT "5 ";: IF B(5) = 1 THEN INVERSE
3690 PRINT "DISABLE TYPE AHEAD BUFFER": NORMAL
3700 VTAB 15: HTAB 2: PRINT "6 ";: IF B(6) = 1 THEN INVERSE
3710 PRINT "DISABLE KEYBOARD EDITING OF MACROS": NORMAL
3720 VTAB 17: HTAB 2: PRINT "7 ";: IF B(7) = 1 THEN INVERSE
3730 PRINT "LOCK OUT AUTO DOWN LOAD": NORMAL
3740 ST = 0: FOR P = 1 TO 7:ST = 2 * ST + B(8 - P): NEXT
3800 VTAB 24: HTAB 5: PRINT "SELECT OPTION (RET TO CONTINUE)": GOSUB 90

```

```

0
3810 IF GC$ = CHR$ (13) THEN 14000
3820 IF GC$ = CHR$ (27) THEN 1000
3830 IF GC$ < "8" AND GC$ > "0" THEN B( ASC (GC$) - 48) = 1 - B( ASC (GC
$) - 48)
3840 GOTO 3600
4000 REM

* DISPLAY MACROS *

4010 HOME :: HTAB 14: INVERSE : PRINT "DISPLAY MACROS": NORMAL : POKE 34
,4
4020 PRINT "MODE KY MACRO DESCRIPTION": PRINT
4030 I = 1
4040 IF BU > = 511 THEN GOSUB 19100: GOTO 1000
4050 PRINT :MO = PEEK (AD + I): GOSUB 700: HTAB 7: IF MO < 8 THEN CHR =
LB( PEEK (AD + I + 1)): GOSUB 40: HTAB 10: GOTO 4060
4055 CHR = KA( PEEK (AD + I + 1)): GOSUB 40: HTAB 10
4060 I = I + 2
4070 CHR = PEEK (AD + I) - 128: IF CHR < 0 THEN 4050
4080 IF NOT PEEK (36) THEN HTAB 10
4085 GOSUB 40
4090 I = I + 1: IF BU + I = 512 THEN POKE 32,0: POKE 33,40: PRINT : GOSUB
19100: GOTO 1000
4100 KY = PEEK ( - 16384): IF KY = 155 THEN POKE - 16368,0: GOTO 1000
4110 IF KY > 127 THEN POKE - 16368,0: WAIT - 16384,128: POKE - 16368
,0
4120 GOTO 4070
5000 REM

* EDIT MACRO ENTRY POINT *

5010 INVERSE : HTAB 1: VTAB 1: PRINT "USED: FREE:
": NORMAL : VTAB 5
5020 HOME : HTAB 15: INVERSE : PRINT "EDIT MACRO": NORMAL : POKE 34,3: PRINT
: PRINT
5100 PRINT : VTAB 5: PRINT "MODE:":NAZ(0) = - 2: GOSUB 8180: GOSUB 300
5110 VTAB 5: HTAB 6: GOSUB 900: IF ASC (GC$) > 95 THEN GC$ = CHR$ ( ASC
(GC$) - 32)
5120 IF GC$ = "L" THEN AL = 1 - AL:KP = 0: GOSUB 300
5130 IF GC$ = "C" THEN CN = 1 - CN:KP = 0: GOSUB 300
5140 IF GC$ = "S" THEN SH = 1 - SH:KP = 0: GOSUB 300
5150 IF GC$ = "K" THEN KP = 1 - KP:CN = 0:AL = 0:SH = 0: GOSUB 300
5160 IF GC$ = CHR$ (27) THEN 1000
5170 IF GC$ = CHR$ (13) THEN 5200
5180 GOTO 5110
5200 HTAB 1: VTAB 7: PRINT "ENTER MACRO ACCESS KEY: "; GET GC$: IF GC$
= "" THEN KY = 0: GOTO 5204
5201 IF ASC (GC$) > 95 THEN GC$ = CHR$ ( ASC (GC$) - 32)
5203 KY = ASC (GC$):CHR = KY: GOSUB 40
5204 MK = LF(KY): IF KP THEN GOSUB 800
5206 IF MK < 0 THEN PRINT CHR$ (8);" ": GOTO 5200
5208 PRINT : VTAB 20: HTAB 16: FLASH : PRINT "WORKING"
5210 GOSUB 100: IF FA = 0 THEN GOSUB 400: GOSUB 200: GOTO 5500
5220 NAZ(0) = 0
5500 VTAB 9: CALL - 958: NORMAL : GOSUB 8000
5510 IF NAZ(0) < = 0 THEN 1000

```

\* EDIT MACROS HERE \*

**\* DISK LOAD ENTRY POINT \***

**\* DISK SAVE ENTRY POINT \***

Page X-27

```

14050 POKE AD - 2,ST
14100 PRINT D$"BSAVE"NA$,A$8CAO,L$2F0"
14999 GOTO 1000: REM * MAIN MENU *
18000 REM

```

\*\*\* Getput for strings \*\*\*

```

18020 DU$ = "":DD$ = "": PRINT NA$;
18030 CALL 769:DU$ = CHR$ ( PEEK (768) - 128): IF ASC (DU$) = 13 THEN
    PRINT CHR$ (29);: CALL - 868: GOTO 18100
18040 IF DU$ = CHR$ (3) OR DU$ = CHR$ (27) THEN NA$ = "": POP : GOTO 1
    000
18050 IF ASC (DU$) = 8 AND LEN (NA$) THEN DD$ = RIGHT$ (NA$,1) + DD$:
    NA$ = MID$ (NA$,1, LEN (NA$) - 1): PRINT CHR$ (8)" " CHR$ (8);: GOTO
    18030
18060 IF ASC (DU$) = 21 AND LEN (DD$) THEN DU$ = LEFT$ (DD$,1)
18070 IF ASC (DU$) = 24 AND LEN (NA$) THEN FOR DU = 1 TO LEN (NA$): PRINT
    CHR$ (8)" " CHR$ (8);: NEXT :DD$ = "":NA$ = "": GOTO 18030
18080 IF ASC (DU$) < 32 GOTO 18030
18090 PRINT DU$;NA$ = NA$ + DU$:DD$ = MID$ (DD$,2, LEN (DD$) - ( LEN (
    DD$) > 0)): GOTO 18030
18100 RETURN
19000 REM

```

\* MISC SUBROUTINES \*

```

19100 VTAB 24: HTAB 10: PRINT "HIT ANY KEY TO CONTINUE";: GOSUB 900: RETURN

19500 REM * ONERR ENTRY POINT *
19540 ER = PEEK (222)
19550 CALL 966: TEXT : HOME : VTAB 12
19600 ON ER GOTO 19700,19700,19700,19620,19700,19630,19700,19640,19650,1
    9660,19670
19610 IF ER = 255 THEN TEXT : HOME : VTAB 12: HTAB 9: PRINT "CONTROL-C
    INTERRUPT ERROR": GOSUB 19100: GOTO 1000
19615 IF ER > = 12 THEN 19700
19620 HTAB 09: PRINT "WRITE PROTECTED DISK ERROR": GOSUB 19100: GOTO 100
    0
19630 HTAB 12: PRINT "FILE NOT FOUND ERROR": GOSUB 19100:NA$ = "": GOTO
    1000
19640 HTAB 14: PRINT "DISK I/O ERROR": GOSUB 19100: GOTO 1000
19650 HTAB 14: PRINT "DISK FULL ERROR": GOSUB 19100: GOTO 1000
19660 HTAB 11: PRINT "DISK FILE LOCKED ERROR": GOSUB 19100: GOTO 1000
19670 HTAB 13: PRINT "DOS SYNTAX ERROR": GOSUB 19100: GOTO 1000
19700 HTAB 11: PRINT "ERROR PEEK(222)==>";ER: GOSUB 19100
19750 GOTO 1000: REM * MAIN MENU *
25000 REM * INITIALIZE PROGRAM VARIABLES *
25010 PR# 0: IN# 0: POKE - 16296,0: CALL 1002: PRINT CHR$ (4)"NOMONIOC
    ": TEXT : HOME : VTAB 12: HTAB 16: FLASH : PRINT "WORKING": NORMAL
25020 PRINT CHR$ (4)"BLOAD DOWNLOAD,A$8CAO"
25130 DIM LF(96): REM * LOOKUP TABLE FORWARD *
25140 DIM LB(50): REM * LOOKUP TABLE BACKWARD *
25150 DIM NA$(512): REM * EDIT ARRAY *
25180 DIM KA(15): REM * KEYPAD LOOKUP TABLE *
25300 FP = 1:AP = 1:CN = 0:AL = 0:KP = 0:SH = 0:AD = 36098:ST = 0:BU = 51

```

1

```

25310 QA$(1) = "C":QA$(2) = "S":QA$(3) = "L":QA$(4) = "K":D$ = CHR$(4)
25320 ONERR GOTO 19500
25400 POKE 966,104: POKE 967,168: POKE 968,104: POKE 969,166: POKE 970,2
23: POKE 971,154: POKE 972,72: POKE 973,152: POKE 974,72: POKE 975,9
6
25410 POKE 769,32: POKE 770,12: POKE 771,253: POKE 772,141: POKE 773,0: POKE
774,3: POKE 775,96
25500 DATA -1,-1,-1,-1,-1,-1,-1,-1,28,-1,-1,-1,-1,49,-1,-1,-1,-1,-1,-1,-1,
1,29,-1,-1,-1,-1,-1,43,-1,-1,-1,-1
25520 DATA 45,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,37,9,38,39,7,42,41,0,1,2
,3,4,5,6,8,27,-1,-1,-1,-1
25540 DATA 48,44,34,32,20,12,21,22,23,17,24,25,26,36,35,18,19,10,13,40,
14,16,33,11,31,15,30,-1,-1,-1,-1,-1
25550 DATA 51,52,53,54,55,56,57,48,58,45,81,87,69,82,84,89,85,73,79,80,6
8,70,71,72,74,75,76,59,8,21,90,88,67,86,66,78,77,44,46,47,83,50,49,2
7,65,32,0,0,0,13
25560 DATA 48,52,56,43,49,53,57,45,50,54,46,13,51,55,44,0
25600 REM * READ IN DATA STATEMENTS *
25610 FOR I = 0 TO 95: READ LF(I): NEXT
25620 FOR I = 0 TO 49: READ LB(I): NEXT
25630 FOR I = 0 TO 15: READ KA(I): NEXT
25999 GOTO 1000: REM * MAIN MENU *
30000 REM * CN - CONTROL KEY SELECT
30010 REM * AL - ALPHA LOCK SELECT
30020 REM * SH - SHIFT KEY SELECT
30030 REM * KP - KEYPAD SELECT
30040 REM * LF - LOOKUP TABLE FORWARD
30050 REM * LB - LOOKUP TABLE BACKWARD
30060 REM * KA - KEYPAD LOOKUP TABLE
30070 REM * EA - EDIT ARRAY
30080 REM * MK - MACRO KEY
30090 REM * AP - ARRAY POINTER
30110 REM * GC - GET CHARACTER
30120 REM * MO - MODE OF MACRO
30130 REM * BU - BUFFER SPACE AVAILABLE
30140 REM * NA - NAME OF DISK FILE
30150 REM * ER - ERROR NUMBER
30160 REM * FA - FAIL (SEARCH) FLAG
30170 REM * ST - STATUS OF DOWNLOAD OPTIONS
30180 REM * DU - DUMMY VARIABLE
30190 REM * DD - DOUBLE DUMMY VARIABLE
40000 REM *****
40010 REM * *
40020 REM * ENHANCER ][ MACRO EDITOR *
40030 REM * *
40040 REM * NOVEMBER 4, 1981 *
40050 REM * *
40070 REM *****

```

1

## Appendix X: Supporting Software

This page has been purposely left (almost) blank.

## Appendix X: Supporting Software

### Section X.e The Down Load Program

The Macro Editor [section: X.d] creates BRUNable programs which are down load programs. The Macro Editor loads in the file named DOWNLOAD each time it runs. This file should NOT be deleted from the disc. Any file output from the Macro Editor may be put onto any disc and BRUN. Doing so will cause keyboard macros to be downloaded from disc.

```

6 *****
7 *
8 *          ENHANCER ] [
9 *
10 *      MACRO DOWNLOAD PROGRAM
11 *
12 *      11/8/1981      19:30
13 *
14 *****
15 *
16 MACTABLE EQU $8D00
17 KBD      EQU $C000
18 KBDSTRB  EQU $C010
19 WAIT     EQU $FCA8
20 BASEL    EQU $00
21 BASEH    EQU $01
22 *
23          ORG $8CA0
24          OBJ $8CA0
25 *
8CA0: AD 5E C0 26 WRMACROS LDA $C05E      ; SEND START DOWNLOAD SIGNAL
8CA3: A9 8D 27          LDA #>MACTABLE ; INITIALIZE POINTERS
8CA5: 85 01 28          STA BASEH
8CA7: A0 00 29          LDY #$00
8CA9: 84 00 30          STY BASEL
8CAB: B1 00 31 WMLoop LDA (BASEL),Y    ; GET BYTE FROM TABLE
8CAD: 8D F5 8C 32          STA BYTE    ; SAVE IT FOR ROTATING
8CB0: 20 C3 8C 33          JSR WRBYTE  ; OUTPUT BYTE TO ENHANCER
8CB3: C8 34          INY              ; ADVANCE POINTER
8CB4: D0 02 35          BNE WMSKIP
8CB6: E6 01 36          INC BASEH
8CB8: C0 03 37 WMSKIP  CPY #$03      ; CONTINUE UNTIL 515 BYTES ARE SENT
8CBA: D0 EF 38          BNE WMLoop
8CBC: A5 01 39          LDA BASEH
8CBE: C9 8F 40          CMP #>MACTABLE+$200
8CC0: D0 E9 41          BNE WMLoop
8CC2: 60 42          RTS
43 *
8CC3: 8C F4 8C 44 WRBYTE  STY YSAVE      ; SAVE Y REGISTER
8CC6: A0 08 45          LDY #$08      ; LOOP 8 TIMES
8CC8: 0E F5 8C 46 BYTELoop ASL BYTE    ; SHIFT BIT INTO CARRY
8CCB: 20 D5 8C 47          JSR WRBIT   ; OUTPUT BIT TO ENHANCER
8CCE: 88 48          DEY
8CCF: D0 F7 49          BNE BYTELoop  ; CONTINUE LOOP
8CD1: AC F4 8C 50          LDY YSAVE   ; RECOVER Y REGISTER
8CD4: 60 51          RTS
52 *
8CD5: 08 53          WRBIT  PHP        ; SAVE CARRY
8CD6: AD 00 C0 54 WBLoop  LDA KBD      ; WAIT FOR RUBOUT FROM ENHANCER
8CD9: 10 FB 55          BPL WBLoop
8CDB: C9 FF 56          CMP #$FF
8CDD: D0 F7 57          BNE WBLoop
8CDF: AD 5F C0 58          LDA $C05F   ; TURN OFF THE START SIGNAL
8CE2: 28 59          PLP              ; RECOVER CARRY
8CE3: 90 03 60          BCC ZEROBIT   ; IF CLEAR DELAY ONCE
8CE5: 20 EF 8C 61          JSR DELAY  ; DELAY TWICE
8CE8: 20 EF 8C 62 ZEROBIT JSR DELAY
8CEB: 2C 10 C0 63          BIT KBDSTRB ; CLEAR KEY STROBE (ACKNOWLEDGE)

```



```

8CEE: 60      64      RTS
           65      *
8CEF: A9 08   66      DELAY      LDA  #08
8CF1: 4C A8 FC 67      JMP      WAIT
           68      *
8CF4: 00      69      YSAVE      HEX  00
8CF5: 00      70      BYTE      HEX  00

```

--END ASSEMBLY--

ERRORS: 0

86 BYTES

# SYMBOL TABLE - ALPHABETICAL ORDER:

|         |         |        |         |            |         |          |         |
|---------|---------|--------|---------|------------|---------|----------|---------|
| BASEH   | =\$01   | BASEL  | =\$00   | BYTE       | =\$8CF5 | BYTELOOP | =\$8CC8 |
| DELAY   | =\$8CEF | KBD    | =\$C000 | KBDSTRB    | =\$C010 | MACTABLE | =\$8D00 |
| WAIT    | =\$FCA8 | WBLOOP | =\$8CD6 | WMLOOP     | =\$8CAB | WMSKIP   | =\$8CB8 |
| WRBIT   | =\$8CD5 | WRBYTE | =\$8CC3 | ? WRMACROS | =\$8CA0 | YSAVE    | =\$8CF4 |
| ZEROBIT | =\$8CE8 |        |         |            |         |          |         |

# SYMBOL TABLE - NUMERICAL ORDER:

|        |         |          |         |            |         |         |         |
|--------|---------|----------|---------|------------|---------|---------|---------|
| BASEL  | =\$00   | BASEH    | =\$01   | ? WRMACROS | =\$8CA0 | WMLOOP  | =\$8CAB |
| WMSKIP | =\$8CB8 | WRBYTE   | =\$8CC3 | BYTELOOP   | =\$8CC8 | WRBIT   | =\$8CD5 |
| WBLOOP | =\$8CD6 | ZEROBIT  | =\$8CE8 | DELAY      | =\$8CEF | YSAVE   | =\$8CF4 |
| BYTE   | =\$8CF5 | MACTABLE | =\$8D00 | KBD        | =\$C000 | KBDSTRB | =\$C010 |
| WAIT   | =\$FCA8 |          |         |            |         |         |         |

## Appendix X: Supporting Software

### Section X.f The OUTPATCH (Pascal) Program

Note: OUTPATCH was formerly called KEYPATCH.

OUTPATCH patches SYSTEM.APPLE of Pascal to allow the display of lower case characters. Load a copy of SYSTEM.APPLE onto your disc and Execute OUTPATCH. You may then use this SYSTEM.APPLE file on your Pascal Discs. ALWAYS MAKE BACKUP COPIES OF YOUR ORIGINAL FILES!!!

Note: Pascal is contained on the back side of the Enhancer ][ Utilities Disc. This side is write protected. It is not intended to be used more than a few times. Not all the tracks have been initialized, therefore, some bad blocks are likely to be found on this side of the disc.

## Appendix: Supporting Software

### Listing of:

PROGRAM OUTPATCH;

```
{ This program patches the SYSTEM.APPLE for displaying }  
{ lower case with the KEYBOARD & DISPLAY ENHANCER for }  
{ Pascal 1.1.                      Darrell Aldrich 1/81 }
```

VAR BUF:PACKED ARRAY [0..31,0..511] OF 0..255;

F:FILE;

I:INTEGER;

BEGIN

RESET(F,'#4:SYSTEM.APPLE');

I:=BLOCKREAD (F,BUF,32);

CLOSE(F);

BUF[5,388]:=76; BUF[5,389]:=156; BUF[5,390]:=219;

BUF[5,391]:=177; BUF[5,392]:=240; BUF[5,393]:=76;

BUF[5,394]:=142; BUF[5,395]:=219; BUF[5,396]:=177;

BUF[5,397]:=242; BUF[5,398]:=72; BUF[5,399]:=41;

BUF[5,400]:=127; BUF[5,401]:=201; BUF[5,402]:=64;

BUF[5,403]:=104; BUF[5,404]:=144; BUF[5,405]:=3;

BUF[5,406]:=73; BUF[5,407]:=160; BUF[5,408]:=96;

BUF[5,409]:=73; BUF[5,410]:=128; BUF[5,411]:=96;

BUF[5,428]:=32; BUF[5,429]:=135; BUF[5,430]:=219;

BUF[5,431]:=234; BUF[5,448]:=32; BUF[5,449]:=140;

BUF[5,450]:=219; BUF[5,451]:=234; BUF[5,169]:=176;

BUF[5,170]:=4; BUF[5,171]:=234; BUF[5,172]:=234;

RESET(F,'#4:SYSTEM.APPLE');

I:=BLOCKWRITE(F,BUF,32);

CLOSE(F);

END.

```

6 *****
7 *
8 * BASIC QUICK LOADER *
9 *
10 * CONTRIBUTED BY RON ALDRICH *
11 *
12 * 8/28/1981 *
13 *
14 *****
15 *
16 TEMP EQU $1E
17 IOB EQU $48
18 *
19 READ EQU $01
20 VOLUME EQU $03
21 TRACK EQU $04
22 SECTOR EQU $05
23 BUFFER EQU $08
24 COMMAND EQU $0C
25 *
26 RWTS EQU $03D9
27 GETFMP EQU $03DC
28 GETIOB EQU $03E3
29 *
30 DEST EQU $D000
31 *
32 ORG $0300
33 OBJ $D000
34 *
35 *****
36 *
37 * DISK SET UP *
38 *
39 *****
40 *
0300: 2C 83 C0 41 BIT $C083
0303: 2C 83 C0 42 BIT $C083
0306: 8A 43 TXA
0307: 48 44 PHA
0308: 20 DC 03 45 JSR GETFMP ; GET LAST USED FILE MGR PRAM LIST
030B: 84 1E 46 STY TEMP
030D: 85 1F 47 STA TEMP+$01
030F: A0 0E 48 LDY #$0E ; FIND TRACK SECTOR LIST OF FILE
0311: B1 1E 49 LDA (TEMP),Y
0313: 48 50 PHA
0314: C8 51 INY
0315: B1 1E 52 LDA (TEMP),Y
0317: 85 1F 53 STA TEMP+$01
0319: 68 54 PLA
031A: 85 1E 55 STA TEMP
031C: 20 E3 03 56 JSR GETIOB ; GET DOS'S I/O BLOCK
031F: 84 48 57 STY IOB
0321: 85 49 58 STA IOB+$01
0323: A0 03 59 LDY #VOLUME ; VOLUME = 0
0325: A9 00 60 LDA #$00
0327: 91 48 61 STA (IOB),Y
0329: A0 0C 62 LDY #COMMAND ; COMMAND = READ
032B: A9 01 63 LDA #READ

```

```

032D: 91 48      64      STA  (IOB),Y
032F: A0 08      65      LDY  #BUFFER      ; BUFFER = DESTINATION OF DATA
0331: A9 00      66      LDA  #<DEST
0333: 91 48      67      STA  (IOB),Y
0335: C8         68      INY
0336: A9 D0      69      LDA  #>DEST
0338: 91 48      70      STA  (IOB),Y
033A: A2 0E      71      LDX  #SOE      ; START READING 2ND T. S. L. PAIR
                                *
                                *****
                                *
                                *
                                READ FILE
                                *
                                *
                                *****
                                *
033C: 8A         79      LOOP      TXA
033D: A8         80      TAY
033E: B1 1E      81      LDA  (TEMP),Y      ; GET TRACK
0340: F0 24      82      BEQ  DONE      ; DONE IF TRACK 0
0342: A0 04      83      LDY  #TRACK
0344: 91 48      84      STA  (IOB),Y
0346: 8A         85      TXA
0347: A         86      TAY
0348: C8         87      INY
0349: B1 1E      88      LDA  (TEMP),Y      ; GET SECTOR
034B: A0 05      89      LDY  #SECTOR
034D: 91 48      90      STA  (IOB),Y
034F: 8A         91      TXA      ; SAVE X
0350: 48         92      PHA
0351: 20 E3 03   93      JSR  GETIOB      ; GET DOS'S I/O BLOCK
0354: 20 D9 03   94      JSR  RWTS      ; READ SECTOR
0357: 68         95      PLA      ; RECOVER XX
0358: AA         96      TAX
0359: E8         97      INX      ; ADD 2
035A: E8         98      INX
035B: A0 09      99      LDY  #BUFFER+$01
035D: B1 48     100      LDA  (IOB),Y      ; INCREMENT HIGH BYTE
035F: 18         101      CLC      ; OF BUFFER ADDRESS
0360: 69 01     102      ADC  #$01
0362: 91 48     103      STA  (IOB),Y
0364: D0 D6     104      BNE  LOOP      ; OFTEN TAKEN
                                *
0366: 2C 81 C0 106      DONE      BIT  $C081
0369: 68         107      PLA
036A: AA         108      TAX
036B: 60         109      RTS

```

--END ASSEMBLY--

ERRORS: 0

108 BYTES

SYMBOL TABLE - ALPHABETICAL ORDER:

|        |       |         |       |      |         |      |         |
|--------|-------|---------|-------|------|---------|------|---------|
| BUFFER | =\$08 | COMMAND | =\$0C | DEST | =\$D000 | DONE | =\$0366 |
|--------|-------|---------|-------|------|---------|------|---------|

|        |         |        |         |        |       |      |         |
|--------|---------|--------|---------|--------|-------|------|---------|
| GETFMP | =\$03DC | GETIOB | =\$03E3 | IOB    | =\$48 | LOOP | =\$033C |
| READ   | =\$01   | RWTS   | =\$03D9 | SECTOR | =\$05 | TEMP | =\$1E   |
| TRACK  | =\$04   | VOLUME | =\$03   |        |       |      |         |

SYMBOL TABLE - NUMERICAL ORDER:

|        |         |         |         |       |         |        |         |
|--------|---------|---------|---------|-------|---------|--------|---------|
| READ   | =\$01   | VOLUME  | =\$03   | TRACK | =\$04   | SECTOR | =\$05   |
| BUFFER | =\$08   | COMMAND | =\$0C   | TEMP  | =\$1E   | IOB    | =\$48   |
| LGOP   | =\$033C | DONE    | =\$0366 | RWTS  | =\$03D9 | GETFMP | =\$03DC |
| GETIOB | =\$03E3 | DEST    | =\$D000 |       |         |        |         |

```

6 *****
7 *
8 * ENHANCER ][ OPERATING SYSTEM *
9 *
10 *
11 * 11 / 3 / 1981 11:00 *
12 *
13 *****
14 *
15 *
16 * C9 C8 C7 C6 C5 C4 C3 C2 C1 C0
17 *
18 *1 3 4 5 6 7 8 9 0 : -
19 *
20 *2 Q W E R T Y U I O P
21 *
22 *3 D F G H J K L ; ^H ^U
23 *
24 *4 Z X C V B N M , . /
25 *
26 *5 S 2 1 ^] A SP ^M
27 *
28 *
29 * C10 - CONTROL
30 * C11 - SHIFT
31 * C12 - REPEAT
32 * C13 - RESET
33 * C14 - ACKNOWLEDGE
34 * C15 - OPTION
35 *
36 *
37 CONTROL EQU $04
38 SHIFT EQU $08
39 REPEAT EQU $10
40 RESET EQU $20
41 ACKNWLG EQU $40
42 OPTION EQU $80
43 *
44 DSHIFTLK EQU $01
45 DMODESEL EQU $02
46 MODESET EQU $04
47 DAUTORPT EQU $08
48 DBUFFER EQU $10
49 DMACDEF EQU $20
50 DDNLOAD EQU $40
51 *
52 CKSUM EQU $00
53 BEGRPT EQU $40
54 STRPT EQU $F0
55 FAST EQU $FB
56 MCDLY EQU $08
57 DLY EQU $10
58 DBTIME EQU $04
59 LKTIME EQU $10
60 *
61 TESTL EQU $00
62 TESTH EQU $01
63 TEMP EQU $02

```

|     |          |     |              |
|-----|----------|-----|--------------|
| 64  | *        |     |              |
| 65  | BUFIN    | EQU | \$00         |
| 66  | BUFOUT   | EQU | \$01         |
| 67  | MAPL     | EQU | \$02         |
| 68  | MAPH     | EQU | \$03         |
| 69  | FLUSH    | EQU | \$04         |
| 70  | LOCKFLG  | EQU | \$05         |
| 71  | HALFLOCK | EQU | \$06         |
| 72  | MACFLG   | EQU | \$07         |
| 73  | REPT     | EQU | \$08         |
| 74  | MTXSAVE  | EQU | \$09         |
| 75  | REPT1    | EQU | \$0A         |
| 76  | DBCNT    | EQU | \$0B         |
| 77  | DBKEY    | EQU | \$0C         |
| 78  | LOCKCNT  | EQU | \$0D         |
| 79  | SPKEYS1  | EQU | \$0E         |
| 80  | MODE1    | EQU | \$0F         |
| 81  | *        |     |              |
| 82  | DEFLAGS  | EQU | \$10         |
| 83  | MODE     | EQU | \$11         |
| 84  | KEY      | EQU | \$12         |
| 85  | BUFMODE  | EQU | \$13         |
| 86  | PWROFF   | EQU | \$14         |
| 87  | SPEED    | EQU | \$15         |
| 88  | CHAR     | EQU | \$16         |
| 89  | SRCHL    | EQU | \$17         |
| 90  | SRCHH    | EQU | \$18         |
| 91  | MOVEL    | EQU | \$19         |
| 92  | MOVEH    | EQU | \$1A         |
| 93  | TENDL    | EQU | \$1B         |
| 94  | TENDH    | EQU | \$1C         |
| 95  | AMODE    | EQU | \$1D         |
| 96  | DFMODE   | EQU | \$1E         |
| 97  | DLFLAG   | EQU | \$1F         |
| 98  | *        |     |              |
| 99  | *        |     |              |
| 100 | *        |     |              |
| 101 | OLDKEY   | EQU | \$20         |
| 102 | *        |     |              |
| 103 | MTXTBL   | EQU | \$40         |
| 104 | *        |     |              |
| 105 | BUFFER   | EQU | \$80         |
| 106 | *        |     |              |
| 107 | MACTABLE | EQU | \$0200       |
| 108 | *        |     |              |
| 109 | MTRIX1   | EQU | \$0A00       |
| 110 | MTRIX2   | EQU | \$0C00       |
| 111 | SPKEYS   | EQU | \$0C01       |
| 112 | *        |     |              |
| 113 | KEYOUT   | EQU | \$0E0D       |
| 114 | *        |     |              |
| 115 | OBJECT   | EQU | \$8800       |
| 116 | EPROM    | EQU | \$1800       |
| 117 | RESVEC   | EQU | EPROM+\$07FC |



```

119 *****
120 *
121 *      KEYBOARD MAPS      *
122 *
123 *****
124 *
125      ORG  EPROM
126      OBJ  OBJECT
127 *

1800: B3 B4 B5
1803: B6 B7 B8
1806: B9 B0 BA
1809: AD      128  NTBL      ASC  "34567890:~"
180A: D1 D7 C5
180D: D2 D4 D9
1810: D5 C9 CF
1813: D0      129          ASC  "QWERTYUIOP"
1814: C4 C6 C7
1817: C8 CA CB
181A: CC BB   130          ASC  "DFGHJKL;"
181C: 88 95   131          HEX  8895
181E: DA D8 C3
1821: D6 C2 CE
1824: CD AC AE
1827: AF      132          ASC  "ZXCVBNM,./"
1828: D3 B2 B1 133          ASC  "S21"
182B: 9B      134          HEX  9B
182C: C1 A0 A0
182F: A0 A0   135          ASC  "A      "
1831: 8D      136          HEX  8D
      137 *
      138      DS      14
      139 *

1840: B3 B4 B5
1843: B6 B7 B8
1846: B9 B0 BA
1849: AD      140  CTBL      ASC  "34567890:~"
184A: 91 97 85
184D: 92 94 99
1850: 95 89 8F
1853: 90      141          HEX  91978592949995898F90
1854: 84 86 87
1857: 88 8A 8B
185A: 8C BB 88
185D: 95      142          HEX  848687888A8B8CBB8895
185E: 9A 98 83
1861: 96 82 8E
1864: 8D      143          HEX  9A988396828E8D
1865: AC AE AF 144          ASC  ",./"
1868: 93      145          HEX  93
1869: B2 B1   146          ASC  "21"
186B: 9B      147          HEX  9B
186C: 81      148          HEX  81
186D: A0 A0 A0
1870: A0      149          ASC  "      "
1871: 8D      150          HEX  8D
      151 *
      152      DS      14

```

|                |     |       |     |                      |
|----------------|-----|-------|-----|----------------------|
| 1880: A3 A4 A5 | 153 | *     |     |                      |
| 1883: A6       | 154 | STBL  | ASC | "#%&"                |
| 1884: A7       | 155 |       | HEX | A7                   |
| 1885: A8 A9 B0 |     |       |     |                      |
| 1888: AA BD    | 156 |       | ASC | "()0*-"              |
| 188A: D1 D7 C5 |     |       |     |                      |
| 188D: D2 D4 D9 |     |       |     |                      |
| 1890: D5 C9 CF |     |       |     |                      |
| 1893: C0       | 157 |       | ASC | "QWERTYUIO@"         |
| 1894: C4 C6 C7 |     |       |     |                      |
| 1897: C8 CA CB |     |       |     |                      |
| 189A: CC AB    | 158 |       | ASC | "DFGHJKL+"           |
| 189C: 88 95    | 159 |       | HEX | 8895                 |
| 189E: DA D8 C3 |     |       |     |                      |
| 18A1: D6 C2 DE |     |       |     |                      |
| 18A4: DD BC BE |     |       |     |                      |
| 18A7: BF       | 160 |       | ASC | "ZXCVB~] <>?"        |
| 18A8: D3       | 161 |       | ASC | "S"                  |
| 18A9: A2       | 162 |       | HEX | A2                   |
| 18AA: A1       | 163 |       | ASC | "I"                  |
| 18AB: 9B       | 164 |       | HEX | 9B                   |
| 18AC: C1 A0 A0 |     |       |     |                      |
| 18AF: A0 A0    | 165 |       | ASC | "A "                 |
| 18B1: 8D       | 166 |       | HEX | 8D                   |
|                | 167 | *     |     |                      |
|                | 168 |       | DS  | 14                   |
|                | 169 | *     |     |                      |
| 18C0: A3 A4 A5 |     |       |     |                      |
| 18C3: A6       | 170 | SCTBL | ASC | "#%&"                |
| 18C4: A7       | 171 |       | HEX | A7                   |
| 18C5: A8 A9 B0 |     |       |     |                      |
| 18C8: AA BD    | 172 |       | ASC | "()0*-"              |
| 18CA: 91 97 85 |     |       |     |                      |
| 18CD: 92 94 99 |     |       |     |                      |
| 18D0: 95 89 8F |     |       |     |                      |
| 18D3: 80       | 173 |       | HEX | 91978592949995898F80 |
| 18D4: 84 86 87 |     |       |     |                      |
| 18D7: 88 8A 8B |     |       |     |                      |
| 18DA: 8C       | 174 |       | HEX | 848687888A8B8C       |
| 18DB: AB       | 175 |       | ASC | "+"                  |
| 18DC: 88 95    | 176 |       | HEX | 8895                 |
| 18DE: 9A 98 83 |     |       |     |                      |
| 18E1: 96 82 9E |     |       |     |                      |
| 18E4: 9D       | 177 |       | HEX | 9A988396829E9D       |
| 18E5: BC BE BF | 178 |       | ASC | "<>?"                |
| 18E8: 93 A2    | 179 |       | HEX | 93A2                 |
| 18EA: A1       | 180 |       | ASC | "I"                  |
| 18EB: 9B       | 181 |       | HEX | 9B                   |
| 18EC: 81       | 182 |       | HEX | 81                   |
| 18ED: A0 A0 A0 |     |       |     |                      |
| 18F0: A0       | 183 |       | ASC | " "                  |
| 18F1: 8D       | 184 |       | HEX | 8D                   |
|                | 185 | *     |     |                      |
|                | 186 |       | DS  | 14                   |
|                | 187 | *     |     |                      |
| 1900: B3 B4 B5 |     |       |     |                      |
| 1903: B6 B7 B8 |     |       |     |                      |

|       |          |     |       |                          |
|-------|----------|-----|-------|--------------------------|
| 1906: | B9 B0 BA |     |       |                          |
| 1909: | AD       | 188 | UNTBL | ASC "34567890:-"         |
| 190A: | F1 F7 E5 |     |       |                          |
| 190D: | F2 F4 F9 |     |       |                          |
| 1910: | F5 E9 EF |     |       |                          |
| 1913: | F0       | 189 |       | ASC "qwertyuiop"         |
| 1914: | E4 E6 E7 |     |       |                          |
| 1917: | E8 EA EB |     |       |                          |
| 191A: | EC BB    | 190 |       | ASC "dfghjkl;,"          |
| 191C: | 88 95    | 191 |       | HEX 8895                 |
| 191E: | FA F8 E3 |     |       |                          |
| 1921: | F6 E2 EE |     |       |                          |
| 1924: | ED AC AE |     |       |                          |
| 1927: | AF       | 192 |       | ASC "zxcvbnm,./"         |
| 1928: | F3 B2 B1 | 193 |       | ASC "s21"                |
| 192B: | 9B       | 194 |       | HEX 9B                   |
| 192C: | E1 A0 A0 |     |       |                          |
| 192F: | A0 A0    | 195 |       | ASC "a "                 |
| 1931: | 8D       | 196 |       | HEX 8D                   |
|       |          | 197 | *     |                          |
|       |          | 198 |       | DS 14                    |
|       |          | 199 | *     |                          |
| 1940: | FF       | 200 | UCTBL | ASC ""                   |
| 1941: | 9C 9D 9E | 201 |       | HEX 9C9D9E               |
| 1944: | E0 FB FD | 202 |       | ASC "{ }"                |
| 1947: | 80 9F    | 203 |       | HEX 809F                 |
| 1949: | DF       | 204 |       | ASC " "                  |
| 194A: | 91 97 85 |     |       |                          |
| 194D: | 92 94 99 |     |       |                          |
| 1950: | 95 89 8F |     |       |                          |
| 1953: | 90       | 205 |       | HEX 91978592949995898F90 |
| 1954: | 84 86 87 |     |       |                          |
| 1957: | 88 8A 8B |     |       |                          |
| 195A: | 8C       | 206 |       | HEX 848687888A8B8C       |
| 195B: | DE       | 207 |       | ASC "~"                  |
| 195C: | 88 95    | 208 |       | HEX 8895                 |
| 195E: | 9A 98 83 |     |       |                          |
| 1961: | 96 82 8E |     |       |                          |
| 1964: | 8D       | 209 |       | HEX 9A988396828E8D       |
| 1965: | DB DD DC | 210 |       | ASC "[ ]\"               |
| 1968: | 93       | 211 |       | HEX 93                   |
| 1969: | FE FC    | 212 |       | ASC "~ "                 |
| 196B: | 9B       | 213 |       | HEX 9B                   |
| 196C: | 81       | 214 |       | HEX 81                   |
| 196D: | A0 A0 A0 |     |       |                          |
| 1970: | A0       | 215 |       | ASC " "                  |
| 1971: | 8D       | 216 |       | HEX 8D                   |
|       |          | 217 | *     |                          |
|       |          | 218 |       | DS 14                    |
|       |          | 219 | *     |                          |
| 1980: | A3 A4 A5 |     |       |                          |
| 1983: | A6       | 220 | USTBL | ASC "#\$%&"              |
| 1984: | A7       | 221 |       | HEX A7                   |
| 1985: | A8 A9 C0 |     |       |                          |
| 1988: | AA BD    | 222 |       | ASC "(@*~"               |
| 198A: | D1 D7 C5 |     |       |                          |
| 198D: | D2 D4 D9 |     |       |                          |
| 1990: | D5 C9 CF |     |       |                          |

|                |     |        |     |                      |
|----------------|-----|--------|-----|----------------------|
| 1993: D0       | 223 |        | ASC | "QWERTYUIOP"         |
| 1994: C4 C6 C7 |     |        |     |                      |
| 1997: C8 CA CB |     |        |     |                      |
| 199A: CC AB    | 224 |        | ASC | "DFGHJKL+"           |
| 199C: 88 95    | 225 |        | HEX | 8895                 |
| 199E: DA D8 C3 |     |        |     |                      |
| 19A1: D6 C2 CE |     |        |     |                      |
| 19A4: CD BC BE |     |        |     |                      |
| 19A7: BF       | 226 |        | ASC | "ZXCVBNM<>?"         |
| 19A8: D3       | 227 |        | ASC | "s"                  |
| 19A9: A2       | 228 |        | HEX | A2                   |
| 19AA: A1       | 229 |        | ASC | "!"                  |
| 19AB: 9B       | 230 |        | HEX | 9B                   |
| 19AC: C1 A0 A0 |     |        |     |                      |
| 19AF: A0 A0    | 231 |        | ASC | "A "                 |
| 19B1: 8D       | 232 |        | HEX | 8D                   |
|                | 233 | *      |     |                      |
|                | 234 |        | DS  | 14                   |
|                | 235 | *      |     |                      |
| 19C0: FF       | 236 | USCTBL | ASC | ""                   |
| 19C1: 9C 9D 9E | 237 |        | HEX | 9C9D9E               |
| 19C4: E0 FB FD | 238 |        | ASC | "*{}"                |
| 19C7: 80 9F    | 239 |        | HEX | 809F                 |
| 19C9: DF       | 240 |        | ASC | " "                  |
| 19CA: 91 97 85 |     |        |     |                      |
| 19CD: 92 94 99 |     |        |     |                      |
| 19D0: 95 89 8F |     |        |     |                      |
| 19D3: 90       | 241 |        | HEX | 91978592949995898F90 |
| 19D4: 84 86 87 |     |        |     |                      |
| 19D7: 88 8A 8B |     |        |     |                      |
| 19DA: 8C       | 242 |        | HEX | 848687888A8B8C       |
| 19DB: DE       | 243 |        | ASC | "~"                  |
| 19DC: 88 95    | 244 |        | HEX | 8895                 |
| 19DE: 9A 98 83 |     |        |     |                      |
| 19E1: 96 82 8E |     |        |     |                      |
| 19E4: 8D       | 245 |        | HEX | 9A988396828E8D       |
| 19E5: DB DD DC | 246 |        | ASC | "[]\"                |
| 19E8: 93       | 247 |        | HEX | 93                   |
| 19E9: FE FC    | 248 |        | ASC | "~ "                 |
| 19EB: 9B       | 249 |        | HEX | 9B                   |
| 19EC: 81       | 250 |        | HEX | 81                   |
| 19ED: A0 A0 A0 |     |        |     |                      |
| 19F0: A0       | 251 |        | ASC | " "                  |
| 19F1: 8D       | 252 |        | HEX | 8D                   |
|                | 253 | *      |     |                      |
|                | 254 |        | DS  | 14                   |
|                | 255 | *      |     |                      |
| 1A00: B0 B4 B8 |     |        |     |                      |
| 1A03: AB       | 256 | PADTBL | ASC | "048+"               |
| 1A04: B1 B5 B9 |     |        |     |                      |
| 1A07: AD       | 257 |        | ASC | "159-"               |
| 1A08: B2 B6 AE | 258 |        | ASC | "26."                |
| 1A0B: 8D       | 259 |        | HEX | 8D                   |
| 1A0C: B3 B7 AC | 260 |        | ASC | "37,"                |
| 1A0F: 80       | 261 |        | HEX | 80                   |

```

263 *****
264 *
265 *      MAIN RESET ENTRY POINT      *
266 *
267 *****
268 *
1A10: D8 269 RESET1 CLD ; BEGIN COLD START SEQUENCE
1A11: 58 270 CLI
1A12: A5 12 271 LDA KEY ; CHECK FOR COLD START
1A14: 49 A5 272 EOR #$A5
1A16: C5 14 273 CMP PWROFF
1A18: D0 03 274 BNE STARTUP ; IF NO MATCH, DO COLD START
275 *
1A1A: 4C 05 1B 276 JMP RESET3
277 *
278 *****
279 *
280 *      ENHANCER RAM TEST      *
281 *
282 *****
283 *
1A1D: A9 A5 284 STARTUP LDA #$A5 ; TEST STORAGE AREAS
1A1F: 85 00 285 STA TESTL
1A21: A9 5A 286 LDA #$5A
1A23: 85 01 287 STA TESTH
1A25: A9 66 288 LDA #$66
1A27: 85 02 289 STA TEMP
1A29: A5 00 290 LDA TESTL
1A2B: C9 A5 291 CMP #$A5
1A2D: D0 0C 292 BNE MERROR
1A2F: A5 01 293 LDA TESTH
1A31: C9 5A 294 CMP #$5A
1A33: D0 06 295 BNE MERROR
1A35: A5 02 296 LDA TEMP
1A37: C9 66 297 CMP #$66
1A39: F0 0D 298 BEQ MSKIP
299 *
1A3B: A0 00 300 MERROR LDY #$00
1A3D: A2 00 301 LDX #$00
1A3F: CA 302 MLOOP DEX
1A40: D0 FD 303 BNE MLOOP
1A42: 88 304 DEY
1A43: D0 FA 305 BNE MLOOP
1A45: 4C A3 1A 306 JMP ERROR
307 *
1A48: A2 01 308 MSKIP LDX #$01 ; 1=WRITE PASS 0=READ PASS
1A4A: A9 03 309 TSTLOOP LDA #$03 ; START AT LOCATION $0003
1A4C: 85 00 310 STA TESTL
1A4E: A9 00 311 LDA #$00
1A50: 85 01 312 STA TESTH
1A52: A8 313 TAY
1A53: A5 00 314 SETLOOP LDA TESTL ; CREATE A VALUE THAT TESTS
1A55: 4A 315 LSR ; DATA BIT ERRORS AND ADDRESS CONFLI
CTS
1A56: 4A 316 LSR
1A57: 4A 317 LSR
1A58: 4A 318 LSR
1A59: 45 00 319 EOR TESTL

```

```

1A5B: 45 01    320      EOR  TESTH
1A5D: 29 0F    321      AND  #$0F
1A5F: 85 02    322      STA  TEMP
1A61: 0A      323      ASL
1A62: 0A      324      ASL
1A63: 0A      325      ASL
1A64: 0A      326      ASL
1A65: 05 02    327      ORA  TEMP
1A67: E0 00    328      CPX  #$00
1A69: D0 06    329      BNE  WRITE      ; READ OF WRITE DATA
1A6B: D1 00    330      CMP  (TESTL),Y
1A6D: D0 CC    331      BNE  MERROR      ; IF BAD, SEND ERROR MESSAGE
1A6F: F0 02    332      BEQ  READ
1A71: 91 00    333  WRITE STA  (TESTL),Y
1A73: E6 00    334  READ  INC  TESTL      ; ADVANCE TO NEXT LOCATION
1A75: D0 DC    335      BNE  SETLOOP
1A77: E6 01    336      INC  TESTH
1A79: A5 01    337      LDA  TESTH
1A7B: C9 04    338      CMP  #$04
1A7D: 90 D4    339      BLT  SETLOOP
1A7F: CA      340      DEX
1A80: F0 C8    341      BEQ  TSTLOOP
342      *
343      *****
344      *
345      *  ENHANCER ROM CHECKSUM TEST  *
346      *
347      *****
348      *
1A82: A9 18    349  CHECKSUM LDA  #>EPROM      ; START AT $1800
1A84: 85 01    350      STA  TESTH
1A86: A9 00    351      LDA  #$00
1A88: 85 00    352      STA  TESTL
1A8A: A8      353      TAY
1A8B: 48      354      PHA
1A8C: 68      355  CSLOOP PLA      ; RECOVERRR SUM
1A8D: 18      356      CLC
1A8E: 71 00    357      ADC  (TESTL),Y      ; ADD TO COUNT
1A90: 48      358      PHA      ; SAVE SUM
1A91: C8      359      INY
1A92: D0 F8    360      BNE  CSLOOP      ; ADVANCE & LOOP UNTIL $2000
1A94: E6 01    361      INC  TESTH
1A96: A5 01    362      LDA  TESTH
1A98: C9 20    363      CMP  #$20
1A9A: D0 F0    364      BNE  CSLOOP
1A9C: 68      365      PLA      ; GET SUM
1A9D: C9 00    366      CMP  #CKSUM      ; IF NOT EQUAL TO CKSUM THEN ERROR
1A9F: F0 31    367      BEQ  CLRTBLS
1AA1: A0 18    368      LDY  #CSMSG-MTMSG ; TRANSMIT CHECKSUM ERROR
369      *
1AA3: B9 B9 1E 370  ERROR LDA  MTMSG,Y
1AA6: F0 16    371      BEQ  HALT
1AA8: 8D 00 0E 372      STA  KEYOUT
1AAB: 09 80    373      ORA  #$80
1AAD: 8D 00 0E 374      STA  KEYOUT
1AB0: 29 7F    375      AND  #$7F
1AB2: 8D 00 0E 376      STA  KEYOUT
1AB5: A2 00    377      LDX  #$00

```

|                |     |         |     |                                      |
|----------------|-----|---------|-----|--------------------------------------|
| 1AB7: CA       | 378 | ERRWAIT | DEX |                                      |
| 1AB8: D0 FD    | 379 |         | BNE | ERRWAIT                              |
| 1ABA: C8       | 380 |         | INY |                                      |
| 1ABB: 4C A3 1A | 381 |         | JMP | ERROR                                |
|                | 382 | *       |     |                                      |
| 1ABE: 58       | 383 | HALT    | CLI |                                      |
| 1ABF: 4C BE 1A | 384 |         | JMP | HALT                                 |
|                | 385 | *       |     |                                      |
| 1AC2: 68       | 386 | IRQ     | PLA |                                      |
| 1AC3: 48       | 387 |         | PHA |                                      |
| 1AC4: 29 10    | 388 |         | AND | #\$10                                |
| 1AC6: D0 05    | 389 |         | BNE | BREAK                                |
| 1AC8: A0 32    | 390 |         | LDY | #IRQMSG-MTMSG                        |
| 1ACA: 4C A3 1A | 391 |         | JMP | ERROR                                |
|                | 392 | *       |     |                                      |
| 1ACD: A0 3D    | 393 | BREAK   | LDY | #BRKMSG-MTMSG                        |
| 1ACF: 4C A3 1A | 394 |         | JMP | ERROR                                |
|                | 395 | *       |     |                                      |
| 1AD2: A2 1B    | 396 | CLRTBLS | LDX | #ROWEND-ROW+1 ; CLEAR LOOK UP TABLES |
| 1AD4: BD 03 1F | 397 | TBLOOP  | LDA | MTXTBL1,X                            |
| 1AD7: 95 40    | 398 |         | STA | MTXTBL,X                             |
| 1AD9: A9 00    | 399 |         | LDA | #\$00                                |
| 1ADB: 95 20    | 400 |         | STA | OLDKEY,X                             |
| 1ADD: CA       | 401 |         | DEX |                                      |
| 1ADE: 10 F4    | 402 |         | BPL | TBLOOP                               |
| 1AEO: 85 10    | 403 |         | STA | DEFLAGS                              |
| 1AE2: AD 01 0C | 404 |         | LDA | SPKEYS                               |
| 1AE5: 29 40    | 405 |         | AND | #ACKNLG                              |
| 1AE7: 85 13    | 406 |         | STA | BUFMODE                              |
| 1AE9: A9 80    | 407 |         | LDA | #\$80                                |
| 1AEB: 85 1F    | 408 |         | STA | DLFLAG                               |
| 1AED: A9 FB    | 409 |         | LDA | #FAST                                |
| 1AEF: 85 15    | 410 |         | STA | SPEED                                |

```

412 *****
413 * *
414 * COLD RESTART *
415 * *
416 *****
417 *
IAF1: A9 00 418 RESET2 LDA #$00
IAF3: 8D 00 02 419 STA MACTABLE
IAF6: 85 1E 420 STA DFMODE
IAF8: 85 1B 421 STA TENDL
IAFA: A9 02 422 LDA #$02
IAFC: 85 1C 423 STA TENDH
IAFE: AD 01 0C 424 RPWAIT LDA SPKEYS
IB01: 29 10 425 AND #REPEAT
IB03: F0 F9 426 BEQ RPWAIT
427 *
428 *****
429 * *
430 * WARM RESTART *
431 * *
432 *****
433 *
IB05: A5 10 434 RESET3 LDA DEFLAGS
IB07: 29 02 435 AND #DMODESEL
IB09: D0 0A 436 BNE RESET4
IB0B: AD 01 0C 437 LDA SPKEYS
IB0E: 29 08 438 AND #SHIFT
IB10: 49 08 439 EOR #SHIFT
IB12: 4A 440 LSR
IB13: 85 1D 441 STA AMODE
442 *
443 *****
444 * *
445 * HOT RESTART *
446 * *
447 *****
448 *
IB15: A2 0F 449 RESET4 LDX #$0F ; HOT RESTART
IB17: A9 00 450 LDA #$00
IB19: 95 00 451 RSLOOP STA $00,X
IB1B: CA 452 DEX
IB1C: 10 FB 453 BPL RSLOOP
IB1E: 9A 454 TXS
IB1F: A9 40 455 LDA #BEGRPT
IB21: 85 08 456 STA REPT

```



```

458 *****
459 *
460 *          SCAN FOR NEW KEY          *
461 *
462 *****
463 *
1B23: 20 1E 1C 464 SCAN      JSR SPECIAL      ; HANDLE SPECIAL KEYS
1B26: A2 00 465          LDX #$00
1B28: 86 09 466          STX MTXSAVE
1B2A: 20 59 1B 467 SCLOOP    JSR RDKEY      ; READ KEYBOARD
1B2D: 48 468          PHA              ; SAVE MATRIX
1B2E: 15 20 469          ORA OLDKEY,X    ; REMOVE OLD KEYS
1B30: 49 FF 470          EOR #$FF
1B32: D0 77 471          BNE DECODE     ; IF NOT 0 THEN DECODE NEW KEY
1B34: 68 472          PLA              ; RECOVER MATRIX
1B35: 49 FF 473          EOR #$FF
1B37: 95 20 474          STA OLDKEY,X    ; ESTABLISH OLD KEYS
1B39: 05 09 475          ORA MTXSAVE     ; ADD TO KEY DOWN CHECK
1B3B: 85 09 476          STA MTXSAVE
1B3D: E8 477          INX
1B3E: E8 478          INX
1B3F: E0 1C 479          CPX #$1C       ; DONE WITH SCAN ?
1B41: 90 E7 480          BLT SCLOOP     ; NO, CONTINUE
1B43: A5 09 481          LDA MTXSAVE     ; IF OLD KEY DOWN THEN AUTO REPEAT
1B45: D0 21 482          BNE REPCHK
1B47: A5 10 483          LDA DEFLAGS    ; CHECK FOR DEFINE MACROS
1B49: 29 20 484          AND #DMACDEF
1B4B: D0 D6 485          BNE SCAN
1B4D: AD 01 0C 486          LDA SPKEYS
1B50: 29 1C 487          AND #SHIFT.CONTROL.REPEAT
1B52: D0 CF 488          BNE SCAN
1B54: 38 489          SEC
1B55: 66 1E 490          ROR DFMODE
1B57: D0 CA 491          BNE SCAN
492 *
493 *****
494 *
495 *          READ KEYBOARD ROW          *
496 *
497 *****
498 *
1B59: A1 40 499 RDKEY      LDA (MTXTBL,X) ; GET MATRIX
1B5B: E0 0A 500          CPX #$0A          ; USE APPROPRIATE MASK
1B5D: 90 08 501          BLT NOMASK
1B5F: E0 14 502          CPX #$14
1B61: B0 02 503          BGE MASK1
1B63: 09 FC 504          ORA #$FC
1B65: 09 F0 505 MASK1    ORA #$F0
1B67: 60 506 NOMASK    RTS

```

```

508 *****
509 *
510 *      PERFORM REPEAT      *
511 *      & AUTO REPEAT      *
512 *
513 *****
514 *
1B68: AD 01 0C 515 REPCHK   LDA   SPKEYS      ; REPEAT KEY DOWN?
1B6B: 29 10      516          AND   #REPEAT
1B6D: F0 09      517          BEQ   KEYREPT    ; YES, DO REPEAT
1B6F: A5 15      518          LDA   SPEED
1B71: C9 F0      519          CMP   #STRPT
1B73: D0 0D      520          BNE   AUTORPT    ; IF FAST THEN AUTO REPEAT
1B75: 4C 23 1B 521          JMP   SCAN
522 *
1B78: A9 F0      523 KEYREPT  LDA   #STRPT
1B7A: C5 08      524          CMP   REPT
1B7C: 90 04      525          BLT   AUTORPT
1B7E: A5 15      526          LDA   SPEED
1B80: 85 08      527          STA   REPT      ; RESTART COUNTER
528 *
1B82: E6 0A      529 AUTORPT  INC   REPT1      ; INCREMENT REPEAT COUNTER
1B84: A5 0A      530          LDA   REPT1
1B86: 4A          531          LSR
1B87: 90 9A      532          BCC   SCAN
1B89: E6 08      533          INC   REPT
1B8B: D0 96      534          BNE   SCAN      ; IF 0 THEN REPEAT LAST KEY
1B8D: A9 F0      535          LDA   #STRPT
1B8F: 85 08      536          STA   REPT
1B91: A5 1E      537          LDA   DFMODE      ; HANDLE MACROS WIERD
1B93: D0 0A      538          BNE   RDMACRO
1B95: 24 07      539          BIT   MACFLG
1B97: 30 0C      540          BMI   RPMACRO
1B99: 20 99 1D 541          JSR   NOMAC1
1B9C: 4C 23 1B 542          JMP   SCAN
543 *
1B9F: 20 ED 1C 544 RDMACRO  JSR   MCRECHK
1BA2: 4C 23 1B 545          JMP   SCAN
546 *
1BA5: 20 5D 1D 547 RPMACRO  JSR   MACRO
1BA8: 4C 23 1B 548          JMP   SCAN

```

```

550 *****
551 * *
552 *      DECODE NEW KEY      *
553 * *
554 *****
555 *
1BAB: A0 01 556 DECODE LDY #$01      ; SHIFT BITS FROM MATRIX
1BAD: C8     557 DCLOOP INY
1BAE: 0A     558 ASL
1BAF: 90 FC  559 BCC DCLOOP
560 *
561 *****
562 * *
563 *      DEBOUNCE KEY      *
564 * *
565 *****
566 *
1BB1: 84 0C  567 STY DBKEY      ; DEBOUNCE KEYBOARD
1BB3: A9 04  568 LDA #DBTIME    ; DBTIME TIMES
1BB5: 85 0B  569 STA DBCNT
1BB7: 20 59 1B 570 DBLOOP JSR RDKEY
1BBA: 15 20  571 ORA OLDKEY,X
1BBC: A0 01  572 LDY #$01
1BBE: C8     573 DBLOOP1 INY
1BBF: C4 0C  574 CPY DBKEY
1BC1: F0 04  575 BEQ DBEXIT
1BC3: 0A     576 ASL
1BC4: 4C BE 1B 577 JMP DBLOOP1
578 *
1BC7: 0A     579 DBEXIT ASL
1BC8: 90 04  580 BCC GOODKEY    ; GOOD IF KEY STILL DOWN
1BCA: 68     581 PLA
1BCB: 4C 23 1B 582 JMP SCAN      ; GIVE UP
583 *
1BCE: C6 0B  584 GOODKEY DEC DBCNT
1BD0: D0 E5  585 BNE DBLOOP    ; CONTINUE DEBOUNCING UNTIL DONE
586 *
1BD2: 68     587 NODB PLA      ; RECOVER MATRIX
1BD3: 49 FF  588 EOR #$FF
1BD5: 95 20  589 STA OLDKEY,X  ; ESTABLISH OLD KEY
1BD7: A9 80  590 LDA #$80      ; CLEAR HALFLOCK FLAG
1BD9: 85 06  591 STA HALFLOCK
1BDB: A9 40  592 LDA #BEGRPT   ; RESET REPEAT COUNT
1BDD: 85 08  593 STA REPT
1BDF: 98     594 TYA          ; COMPUTE KEY INDEX
1BE0: 18     595 CLC
1BE1: 7D 1F 1F 596 ADC ROW,X
1BE4: 85 16  597 STA CHAR      ; SAVE IN CHAR
598 *
599 *****
600 * *
601 *      COMPUTE MODE      *
602 * *
603 *****
604 *
1BE6: A5 0E  605 LDA SPKEYS1    ; COMPUTE MODE
1BE8: 29 0C  606 AND #SHIFT.CONTROL
1BEA: 4A     607 LSR

```

|                |     |         |               |                              |
|----------------|-----|---------|---------------|------------------------------|
| 1BEB: 4A       | 608 |         | LSR           |                              |
| 1BEC: 05 1D    | 609 |         | ORA AMODE     |                              |
| 1BEE: 85 0F    | 610 |         | STA MODE1     |                              |
|                | 611 | *       |               |                              |
| 1BF0: A5 10    | 612 |         | LDA DEFLAGS   |                              |
| 1BF2: 29 01    | 613 |         | AND #DSHIFTLK |                              |
| 1BF4: 85 02    | 614 |         | STA TEMP      |                              |
| 1BF6: A5 1D    | 615 |         | LDA AMODE     |                              |
| 1BF8: 49 04    | 616 |         | EOR #\$04     |                              |
| 1BFA: 05 02    | 617 |         | ORA TEMP      |                              |
| 1BFC: 05 1E    | 618 |         | ORA DFMODE    |                              |
| 1BFE: F0 04    | 619 |         | BEQ MDSET1    |                              |
| 1C00: A9 00    | 620 |         | LDA #\$00     |                              |
| 1C02: F0 06    | 621 |         | BEQ ALOCK     |                              |
|                | 622 | *       |               |                              |
| 1C04: A5 05    | 623 | MDSET1  | LDA LOCKFLG   |                              |
| 1C06: 29 08    | 624 |         | AND #SHIFT    |                              |
| 1C08: 4A       | 625 |         | LSR           |                              |
| 1C09: 4A       | 626 |         | LSR           |                              |
| 1C0A: 05 0F    | 627 | ALOCK   | ORA MODE1     |                              |
| 1C0C: 85 11    | 628 |         | STA MODE      |                              |
|                | 629 | *       |               |                              |
| 1C0E: E0 14    | 630 |         | CPX #\$14     | ; IF FROM KEYPAD CHANGE MODE |
| 1C10: 90 06    | 631 |         | BLT NTKYPAD   |                              |
| 1C12: A9 08    | 632 |         | LDA #\$08     |                              |
| 1C14: 85 11    | 633 |         | STA MODE      |                              |
| 1C16: 85 0F    | 634 |         | STA MODE1     |                              |
| 1C18: 20 CC 1C | 635 | NTKYPAD | JSR GETKEY    | ; PUT KEY IN BUFFER          |
| 1C1B: 4C 23 1B | 636 |         | JMP SCAN      | ; RETURN TO SCAN             |

```

638 *****
639 *
640 *          HANDLE SPECIAL KEYS          *
641 *
642 *          SHIFT CTRL REPT RESET        *
643 *
644 *****
645 *
1C1E: AD 01 OC 646 SPECIAL LDA SPKEYS          ; CREATE SPKEYS1
1C21: 49 FF 647 EOR #$FF
1C23: 85 OE 648 STA SPKEYS1
649 *
650 *****
651 *
652 *          CHECK FOR DOWN LOAD MACROS    *
653 *
654 *****
655 *
1C25: A5 10 656 LDA DEFLAG
1C27: 29 40 657 AND #DDNLOAD          ; CHECK FOR DOWN LOAD DEFEAT
1C29: D0 10 658 BNE NTDNLD
1C2B: A5 OE 659 LDA SPKEYS1
1C2D: 29 80 660 AND #OPTION          ; CHECK FOR AUTO DOWN LOAD
1C2F: F0 0A 661 BEQ NTDNLD
1C31: A5 1F 662 LDA DLFLAG
1C33: D0 0A 663 BNE NTDNLD1
1C35: 38 664 SEC
1C36: 66 1F 665 ROR DLFLAG
1C38: 4C 20 1E 666 JMP DOWNLOAD
1C3B: A9 00 667 NTDNLD LDA #$00
1C3D: 85 1F 668 STA DLFLAG
1C3F: A5 OE 669 NTDNLD1 LDA SPKEYS1
1C41: 29 20 670 AND #RESET          ; CHECK FOR RESET PRESSED
1C43: F0 12 671 BEQ NTRSET
1C45: A5 10 672 LDA DEFLAG
1C47: 29 20 673 AND #DMACDEF        ; CHECK FOR MACRO DEFINE DEFEAT
1C49: D0 09 674 BNE NTRPT
1C4B: A5 OE 675 LDA SPKEYS1
1C4D: 29 10 676 AND #REPEAT        ; CHECK FOR REPEAT-RESET
1C4F: F0 03 677 BEQ NTRPT
1C51: 4C 20 1E 678 JMP DOWNLOAD          ; IF SO, DOWN LOAD
1C54: 4C 05 1B 679 NTRPT JMP RESET3      ; JUST RESET, WARM START
680 *
681 *****
682 *
683 *          CHECK FOR END MACRO DEFINE    *
684 *
685 *****
686 *
1C57: A5 OE 687 NTRSET LDA SPKEYS1
1C59: 29 10 688 AND #REPEAT          ; CHECK FOR REPEAT TO
1C5B: F0 08 689 BEQ NTRPT            ; TERMINATE MACRO DEFINITION
1C5D: A5 1E 690 LDA DFMODE
1C5F: 30 04 691 BMI NTRPT
1C61: A9 00 692 LDA #$00
1C63: 85 1E 693 STA DFMODE

```

```

695 *****
696 * *
697 * PERFORM SHIFT LOCK FUNCTION *
698 * *
699 *****
700 *
1C65: AD 01 OC 701 NTREPT LDA SPKEYS
1C68: 29 08 702 AND #SHIFT
1C6A: 25 05 703 AND LOCKFLG
1C6C: 85 05 704 STA LOCKFLG
705 *
1C6E: A5 0E 706 LDA SPKEYS1
1C70: 29 04 707 AND #CONTROL
1C72: F0 16 708 BEQ LOCKSET
1C74: A5 0D 709 LDA LOCKCNT
1C76: F0 04 710 BEQ GOODLK
1C78: C6 0D 711 DEC LOCKCNT
1C7A: D0 1E 712 BNE OUTPUT
1C7C: A5 05 713 GOODLK LDA LOCKFLG
1C7E: D0 1A 714 BNE OUTPUT
1C80: 24 06 715 BIT HALFLOCK
1C82: 30 16 716 BMI OUTPUT
1C84: A9 08 717 LDA #SHIFT
1C86: 85 06 718 STA HALFLOCK
1C88: D0 10 719 BNE OUTPUT
720 *
1C8A: A5 05 721 LOCKSET LDA LOCKFLG
1C8C: 05 06 722 ORA HALFLOCK
1C8E: 29 08 723 AND #SHIFT
1C90: 85 05 724 STA LOCKFLG
1C92: A9 00 725 LDA #$00
1C94: 85 06 726 STA HALFLOCK
1C96: A9 10 727 LDA #LKTIME
1C98: 85 0D 728 STA LOCKCNT
729 *
730 *****
731 * *
732 * OUTPUT KEY IF POSSIBLE *
733 * *
734 *****
735 *
1C9A: A5 10 736 OUTPUT LDA DEFLAGS
1C9C: 29 10 737 AND #DBUFFER ; CHECK FOR BUFFER DEFEAT
1C9E: 05 13 738 ORA BUFSIZE
1CA0: D0 0B 739 BNE NOBUFF
1CA2: A5 04 740 LDA FLUSH ; NO BUFFER IF FLUSHED
1CA4: F0 07 741 BEQ NOBUFF
1CA6: AD 01 OC 742 LDA SPKEYS
1CA9: 29 40 743 AND #ACKNWLG ; WAIT FOR ACKNOWLEDGE
1CAB: D0 1E 744 BNE SPEXIT
1CAD: A6 01 745 NOBUFF LDX BUFOUT
1CAF: E4 00 746 CPX BUFIN ; BUFFER EMPTY?
1CB1: F0 18 747 BEQ SPEXIT ; YES, EXIT
1CB3: E8 748 INX
1CB4: 8A 749 TXA
1CB5: 29 7F 750 AND #$7F
1CB7: 85 01 751 STA BUFOUT ; 128 CHAR BUFFER
1CB9: AA 752 TAX

```

|                |     |              |                             |
|----------------|-----|--------------|-----------------------------|
| 1CBA: B5 80    | 753 | LDA BUFFER,X | ; GET CHARACTER FROM BUFFER |
| 1CBC: 8D 00 0E | 754 | STA KEYOUT   | ; OUTPUT CHARACTER          |
| 1CBF: 09 80    | 755 | ORA #\$80    |                             |
| 1CC1: 85 04    | 756 | STA FLUSH    |                             |
| 1CC3: 8D 00 0E | 757 | STA KEYOUT   |                             |
| 1CC6: 29 7F    | 758 | AND #\$7F    |                             |
| 1CC8: 8D 00 0E | 759 | STA KEYOUT   |                             |
| 1CCB: 60       | 760 | SPEXIT       | RTS                         |

```

762 *****
763 *
764 *          PROCESS KEY          *
765 *
766 *****
767 *
1CCC: 46 07 768 GETKEY   LSR   MACFLG      ; CLEAR MACRO FLAG
1CCE: A5 11 769         LDA   MODE       ; COMPUTE ASCII CHARACTER
1CD0: 4A     770         LSR
1CD1: 6A     771         ROR
1CD2: 48     772         PHA
1CD3: 6A     773         ROR
1CD4: 29 C0 774         AND   #$C0
1CD6: 85 02 775         STA   MAPL
1CD8: 68     776         PLA
1CD9: 29 03 777         AND   #$03
1CDB: 09 18 778         ORA   #>EPROM
1CDD: 85 03 779         STA   MAPH
1CDF: A4 16 780         LDY   CHAR
1CE1: B1 02 781         LDA   (MAPL),Y
1CE3: 85 12 782         STA   KEY
1CE5: 49 A5 783         EOR   #$A5
1CE7: 85 14 784         STA   PWROFF
785 *
1CE9: A5 1E 786         LDA   DFMODE      ; IS A MACRO BEING DEFINED?
1CEB: F0 70 787         BEQ   MACRO      ; NO, CHECK FOR MACRO KEY
1CED: 30 1A 788 MCRECHK  BMI   MACREATE    ; IF NEG START DEFINITION
789 *
790 *****
791 *
792 *          DEFINE MACRO          *
793 *
794 *****
795 *
1CEF: A5 12 796 MCDFINE  LDA   KEY
1CF1: A0 00 797         LDY   #$00
1CF3: 91 17 798         STA   (SRCHL),Y    ; SAVE KEY IN TABLE
1CF5: 20 E2 1D 799         JSR   NXTBYTE    ; ADVANCE
1CF8: F0 26 800         BEQ   MCABORT    ; TERMINATE DEFFINE IF FAIL
1CFA: A9 00 801         LDA   #$00      ; SAVE END CHARACTER
1CFC: 91 17 802         STA   (SRCHL),Y
1CFE: A5 17 803         LDA   SRCHL    ; SAVE NEW END POINTER
1D00: 85 1B 804         STA   TENDL
1D02: A5 18 805         LDA   SRCHH
1D04: 85 1C 806         STA   TENDH
1D06: 4C 8B 1D 807         JMP   NOMACRO    ; CONTINUE
808 *
809 *****
810 *
811 *          START MACRO DEFINITION  *
812 *
813 *****
814 *
1D09: 46 1E 815 MCREATE  LSR   DFMODE      ; MAKE DFMODE POSITIVE
1DOB: 20 B0 1D 816         JSR   SEARCH    ; IS KEY A MACRO ALREADY?
1DOE: D0 16 817         BNE   MACRMVME    ; YES, REMOVE IT
1D10: 20 E8 1D 818         JSR   ENDCHK     ; IS THERE ROOM?
1D13: F0 0B 819         BEQ   MCABORT    ; NO, ABORT

```



```

ID15: A5 17      820      LDA  SRCHL      ; MOVE = SRCH
ID17: 85 19      821      STA  MOVEL
ID19: A5 18      822      LDA  SRCHH
ID1B: 85 1A      823      STA  MOVEH
ID1D: 4C 48 1D 824      JMP  NEWMACRO
      825      *
ID20: A9 00      826      MCABORT LDA  #$00      ; ABORT DEFINITION
ID22: 85 1E      827      STA  DFMODE
ID24: F0 65      828      BEQ  NOMACRO
      829      *
ID26: 20 ED 1D 830      MACRMOVE JSR  NXTCHAR      ; FIND END OF MACRO
ID29: D0 06      831      BNE  MOVE      ; IF FOUND, MOVE THE REST DOWN
ID2B: 20 B0 1D 832      JSR  SEARCH      ; FIND MACRO AGAIN
ID2E: 4C 17 1E 833      JMP  ENDSET      ; ESTABLISH NEW END
      834      *
ID31: B1 17      835      MOVE    LDA  (SRCHL),Y      ; MOVE REMAINING MACROS DOWN
ID33: 91 19      836      STA  (MOVEL),Y
ID35: E6 19      837      INC  MOVEL
ID37: D0 02      838      BNE  MSKIP1
ID39: E6 1A      839      INC  MOVEH
ID3B: 20 06 1E 840      MSKIP1 JSR  NXTCHK
ID3E: D0 F1      841      BNE  MOVE
ID40: A5 19      842      LDA  MOVEL
ID42: 85 17      843      STA  SRCHL
ID44: A5 1A      844      LDA  MOVEH
ID46: 85 18      845      STA  SRCHH
      846      *
ID48: A5 11      847      NEWMACRO LDA  MODE      ; SAVE MODE AND CHAR
ID4A: 91 17      848      STA  (SRCHL),Y      ; IN TABLE
ID4C: 20 E2 1D 849      JSR  NXTBYTE
ID4F: F0 CF      850      BEQ  MCABORT
ID51: A5 16      851      LDA  CHAR
ID53: 91 17      852      STA  (SRCHL),Y
ID55: 20 E2 1D 853      JSR  NXTBYTE
ID58: F0 C6      854      BEQ  MCABORT
ID5A: 4C 17 1E 855      JMP  ENDSET      ; ESTABLISH NEW END POINTER

```

```

857 *****
858 *
859 * CHECK KEY FOR MACRO OUTPUT *
860 *
861 *****
862 *
1D5D: 20 B0 1D 863 MACRO JSR SEARCH ; IS KEY A MACRO?
1D60: F0 29 864 BEQ NOMACRO ; NO, OUTPUT NORMAL
1D62: 38 865 SEC
1D63: 66 07 866 ROR MACFLG ; SET MACRO FLAG
1D65: B1 17 867 MACLOOP LDA (SRCHL),Y ; STUFF MACRO INTO BUFFER
1D67: 10 46 868 BPL INDONE ; EXIT WHEN DONE
1D69: 85 12 869 STA KEY
1D6B: 49 A5 870 EOR #$A5
1D6D: 85 14 871 STA PWROFF
1D6F: 20 99 1D 872 JSR NOMAC1 ; STUFF CHARACTER
1D72: A5 10 873 LDA DEFLAGS ; IF NO BUFFER, USE DELAY
1D74: 29 10 874 AND #DBUFFER
1D76: 05 13 875 ORA BUFMODE
1D78: F0 0A 876 BEQ NODLY
1D7A: A0 08 877 LDY #MCDLY
1D7C: A2 00 878 MDLOOP LDX #$00
1D7E: 20 B0 1E 879 JSR DLYLOOP
1D81: 88 880 DEY
1D82: D0 F8 881 BNE MDLOOP
1D84: 20 E2 1D 882 NODLY JSR NXTBYTE ; POINT TO NEXT BYTE
1D87: F0 26 883 BEQ INDONE ; IF FAIL THEN DONE
1D89: D0 DA 884 BNE MACLOOP
885 *
1D8B: A5 12 886 NOMACRO LDA KEY ; IF KEY IS CTRL C THEN
1D8D: C9 83 887 CMP #$83 ; FLUSH BUFFER
1D8F: D0 08 888 BNE NOMAC1
1D91: A9 00 889 LDA #$00
1D93: 85 00 890 STA BUFIN
1D95: 85 01 891 STA BUFOUT
1D97: 85 04 892 STA FLUSH
1D99: 20 9A 1C 893 NOMAC1 JSR OUTPUT ; TRY OUTPUT
1D9C: A6 00 894 LDX BUFIN
1D9E: E8 895 INX
1D9F: 8A 896 TXA
1DA0: 29 7F 897 AND #$7F
1DA2: C5 01 898 CMP BUFOUT ; IF BUFFER FULL
1DA4: F0 F3 899 BEQ NOMAC1 ; LOOP UNTIL AVAILABLE
1DA6: AA 900 TAX
1DA7: A5 12 901 LDA KEY
1DA9: 29 7F 902 AND #$7F
1DAB: 95 80 903 STA BUFFER,X ; STUFF KEY IN BUFFER
1DAD: 86 00 904 STX BUFIN
1DAF: 60 905 INDONE RTS

```

```

907 *****
908 *
909 * SEARCH FOR MACRO *
910 *
911 *****
912 *
1DB0: A9 00 913 SEARCH LDA #$00 ; INIT SEARCH VARIABLES
1DB2: 85 17 914 STA SRCHL
1DB4: A8 915 TAY
1DB5: A9 02 916 LDA #>MACTABLE
1DB7: 85 18 917 STA SRCHH
1DB9: 20 0C 1E 918 JSR LASTCHK ; CHECK FOR NO MACROS
1DBC: F0 18 919 BEQ SHEXIT ; IF SO, EXIT
1DBE: A5 17 920 SHLOOP LDA SRCHL ; MOVE = SRCH
1DC0: 85 19 921 STA MOVEH
1DC2: A5 18 922 LDA SRCHH
1DC4: 85 1A 923 STA MOVEH
1DC6: B1 17 924 LDA (SRCHL),Y ; GET BYTE
1DC8: C5 0F 925 CMP MODE1 ; IS IT MODE?
1DCA: F0 0B 926 BEQ MODFND ; YES, CHECK CHAR
1DCC: 20 E2 1D 927 JSR NXTBYTE ; SKIP A BYTE
1DCF: F0 05 928 BEQ SHEXIT ; IF FAIL, EXIT
1DD1: 20 ED 1D 929 NTCHAR JSR NXTCHAR ; ADVANCE TO NEXT MACRO
1DD4: D0 E8 930 BNE SHLOOP ; LOOP UNLESS FAILURE
1DD6: 60 931 SHEXIT RTS
932 *
1DD7: 20 06 1E 933 MODFND JSR NXTCHK ; ADVANCE TO NEXT BYTE
1DDA: F0 FA 934 BEQ SHEXIT ; IF FAIL, EXIT
1DDC: B1 17 935 LDA (SRCHL),Y ; GET BYTE
1DDE: C5 16 936 CMP CHAR ; IS IT CHAR?
1DE0: D0 EF 937 BNE NTCHAR ; NO, TRY AGAIN
938 *
1DE2: E6 17 939 NXTBYTE INC SRCHL ; INCREMENT SEARCH COUNTER
1DE4: D0 02 940 BNE ENDCHK
1DE6: E6 18 941 INC SRCHH
942 *
1DE8: A5 18 943 ENDCHK LDA SRCHH ; CHECK FOR END OF MEMORY
1DEA: C9 04 944 CMP #$04
1DEC: 60 945 RTS
946 *
1DED: E6 17 947 NXTCHAR INC SRCHL ; INCREMENT SEARCH COUNTER
1DEF: D0 02 948 BNE NCSKIP
1DF1: E6 18 949 INC SRCHH
1DF3: A5 18 950 NCSKIP LDA SRCHH ; CHECK FOR END OF MACROS
1DF5: C5 1C 951 CMP TENDH
1DF7: D0 06 952 BNE NCSKIP1
1DF9: A5 17 953 LDA SRCHL
1DFB: C5 1B 954 CMP TENDL
1DFD: F0 06 955 BEQ RTS2
1DFF: B1 17 956 NCSKIP1 LDA (SRCHL),Y ; GET BYTE
1E01: 30 EA 957 BMI NXTCHAR ; LOOP UNTIL HIGH BIT CLEAR
1E03: A9 80 958 LDA #$80 ; RETURN NO FAIL
1E05: 60 959 RTS2 RTS
960 *
1E06: E6 17 961 NXTCHK INC SRCHL ; INCREMENT SEARCH COUNTER
1E08: D0 02 962 BNE LASTCHK
1EOA: E6 18 963 INC SRCHH
964 *

```

|             |     |         |     |       |                           |
|-------------|-----|---------|-----|-------|---------------------------|
| 1E0C: A5 18 | 965 | LASTCHK | LDA | SRCHH | ; CHECK FOR END OF MACROS |
| 1E0E: C5 1C | 966 |         | CMP | TENDH |                           |
| 1E10: D0 04 | 967 |         | BNE | RTS1  |                           |
| 1E12: A5 17 | 968 |         | LDA | SRCHL |                           |
| 1E14: C5 1B | 969 |         | CMP | TENDL |                           |
| 1E16: 60    | 970 | RTS1    | RTS |       |                           |
|             | 971 | *       |     |       |                           |
| 1E17: A5 19 | 972 | ENDSET  | LDA | MOVEL | ; SET END POINTER TO MOVE |
| 1E19: 85 1B | 973 |         | STA | TENDL |                           |
| 1E1B: A5 1A | 974 |         | LDA | MOVEH |                           |
| 1E1D: 85 1C | 975 |         | STA | TENDH |                           |
| 1E1F: 60    | 976 |         | RTS |       |                           |

```

978 *****
979 *
980 * DOWN LOAD MACROS FROM APPLE *
981 *
982 *****
983 *
1E20: 20 75 1E 984 DOWNLOAD JSR RDBYTE ; READ FIRST BYTE
1E23: F0 34 985 BEQ DLERROR ; EXIT ON FAIL
1E25: A5 02 986 LDA TEMP
1E27: 85 0F 987 STA MODEL
1E29: 20 75 1E 988 JSR RDBYTE ; READ END POINTER
1E2C: F0 2B 989 BEQ DLERROR
1E2E: A5 02 990 LDA TEMP
1E30: 85 1B 991 STA TENDL
1E32: 20 75 1E 992 JSR RDBYTE
1E35: F0 22 993 BEQ DLERROR
1E37: A5 02 994 LDA TEMP
1E39: 85 1C 995 STA TENDH
1E3B: 20 5C 1E 996 JSR RDMACROS ; READ MACROS
1E3E: F0 19 997 BEQ DLERROR ; EXIT ON FAIL
1E40: A5 0F 998 LDA MODEL
1E42: 85 10 999 STA DEFLAGS
1E44: 29 04 1000 AND #MODESET ; SET ALPHA MODE
1E46: 85 1D 1001 STA AMODE
1E48: A5 10 1002 LDA DEFLAGS ; SET DEFEAT FLAGS
1E4A: 29 08 1003 AND #DAUTORPT ; SET AUTO REPEAT SPEED
1E4C: F0 04 1004 BEQ NORMRPT
1E4E: A9 F0 1005 LDA #STRPT
1E50: D0 02 1006 BNE SETSPEED
1E52: A9 FB 1007 NORMRPT LDA #FAST
1E54: 85 15 1008 SETSPEED STA SPEED
1E56: 4C 15 1B 1009 JMP RESET4 ; DO HOT RESTART
1010 *
1E59: 4C F1 1A 1011 DLERROR JMP RESET2 ; DO COLD RESTART
1012 *
1E5C: A9 02 1013 RDMACROS LDA #>MACTABLE ; INIT VARIABLES
1E5E: 85 18 1014 STA SRCHH
1E60: A9 00 1015 LDA #$00
1E62: 85 17 1016 STA SRCHL
1E64: 20 75 1E 1017 RMLoop JSR RDBYTE ; READ BYTE
1E67: F0 0B 1018 BEQ RTS3 ; EXIT ON FAIL
1E69: A5 02 1019 LDA TEMP
1E6B: 91 17 1020 STA (SRCHL),Y ; SAVE BYTE IN MACRO TABLE
1E6D: 20 E2 1D 1021 JSR NXTBYTE ; ADVANCE TABLE POINTER
1E70: D0 F2 1022 BNE RMLoop ; CONTINUE LOOP UNTIL FAIL
1E72: A9 FF 1023 LDA #$FF ; NO FAIL
1E74: 60 1024 RTS3 RTS
1025 *
1E75: A0 08 1026 RDBYTE LDY #$08 ; READ 8 BITS
1E77: 20 84 1E 1027 BYTELOOP JSR RDBIT
1E7A: F0 07 1028 BEQ RTS4 ; EXIT ON FAIL
1E7C: 26 02 1029 ROL TEMP ; ROTATE BIT IN
1E7E: 88 1030 DEY
1E7F: D0 F6 1031 BNE BYTELOOP
1E81: A9 FF 1032 LDA #$FF ; NO FAIL
1E83: 60 1033 RTS4 RTS
1034 *
1E84: A9 7F 1035 RDBIT LDA #$7F ; OUTPUT A RUBOUT

```

|       |    |    |    |      |         |             |                         |
|-------|----|----|----|------|---------|-------------|-------------------------|
| 1E86: | 8D | 00 | 0E | 1036 | STA     | KEYOUT      |                         |
| 1E89: | A9 | FF |    | 1037 | LDA     | #\$FF       |                         |
| 1E8B: | 8D | 00 | 0E | 1038 | STA     | KEYOUT      |                         |
| 1E8E: | A9 | 7F |    | 1039 | LDA     | #\$7F       |                         |
| 1E90: | 8D | 00 | 0E | 1040 | STA     | KEYOUT      |                         |
| 1E93: | 20 | AE | 1E | 1041 | JSR     | DELAY       | ; DELAY ONCE            |
| 1E96: | F0 | 0F |    | 1042 | BEQ     | RTS5        | ; EXIT ON BAD HANDSHAKE |
| 1E98: | 20 | AE | 1E | 1043 | JSR     | DELAY       | ; DELAY ONCE            |
| 1E9B: | F0 | 0B |    | 1044 | BEQ     | ZEROBIT     | ; CLEAR CARRY AND EXIT  |
| 1E9D: | 20 | AE | 1E | 1045 | JSR     | DELAY       | ; DELAY TWICE           |
| 1EA0: | 20 | AE | 1E | 1046 | JSR     | DELAY       |                         |
| 1EA3: | F0 | 05 |    | 1047 | BEQ     | ONEBIT      | ; SET CARRY AND EXIT    |
| 1EA5: | A9 | 00 |    | 1048 | LDA     | #\$00       | ; FAIL                  |
| 1EA7: | 60 |    |    | 1049 | RTS5    | RTS         |                         |
|       |    |    |    | 1050 | *       |             |                         |
| 1EA8: | 18 |    |    | 1051 | ZEROBIT | CLC         |                         |
| 1EA9: | B0 |    |    | 1052 |         | HEX B0      |                         |
| 1EAA: | 38 |    |    | 1053 | ONEBIT  | SEC         |                         |
| 1EAB: | A9 | FF |    | 1054 | LDA     | #\$FF       | ; NO FAIL               |
| 1EAD: | 60 |    |    | 1055 |         | RTS         |                         |
|       |    |    |    | 1056 | *       |             |                         |
| 1EAE: | A2 | 10 |    | 1057 | DELAY   | LDX #DLY    |                         |
| 1EB0: | CA |    |    | 1058 | DLYLOOP | DEX         |                         |
| 1EB1: | D0 | FD |    | 1059 |         | BNE DLYLOOP |                         |
| 1EB3: | AD | 01 | 0C | 1060 | LDA     | SPKEYS      |                         |
| 1EB6: | 29 | 40 |    | 1061 | AND     | #ACKNWLG    |                         |
| 1EB8: | 60 |    |    | 1062 |         | RTS         |                         |

```

1064 *****
1065 *
1066 *      MESSAGES AND TABLES
1067 *
1068 *****
1069 *

1EB9: 45 4E 48
1EBC: 41 4E 43
1EBF: 45 52 20
1EC2: 4D 45 4D
1EC5: 4F 52 59
1EC8: 20 46 41
1ECB: 49 4C 55
1ECE: 52 45      1070 MTMSG      ASC 'ENHANCER MEMORY FAILURE'
1ED0: 00          1071          HEX 00
1ED1: 45 4E 48
1ED4: 41 4E 43
1ED7: 45 52 20
1EDA: 43 48 45
1EDD: 43 4B 53
1EE0: 55 4D 20
1EE3: 46 41 49
1EE6: 4C 55 52
1EE9: 45          1072 CSMSG      ASC 'ENHANCER CHECKSUM FAILURE'
1EEA: 00          1073          HEX 00
1EEB: 49 4E 54
1EEE: 45 52 52
1EF1: 55 50 54
1EF4: 20          1074 IRQMSG      ASC 'INTERRUPT '
1EF5: 00          1075          HEX 00
1EF6: 42 52 45
1EF9: 41 4B 20
1EFC: 45 52 52
1EFF: 4F 52 20   1076 BRKMSG      ASC 'BREAK ERROR '
1F02: 00          1077          HEX 00
1078 *
1F03: 10 0A      1079 MTXTBL1    DA  MTRIX1+$10
1F05: 08 0A      1080          DA  MTRIX1+$08
1F07: 04 0A      1081          DA  MTRIX1+$04
1F09: 02 0A      1082          DA  MTRIX1+$02
1F0B: 01 0A      1083          DA  MTRIX1+$01
1F0D: 10 0C      1084          DA  MTRIX2+$10
1F0F: 08 0C      1085          DA  MTRIX2+$08
1F11: 04 0C      1086          DA  MTRIX2+$04
1F13: 02 0C      1087          DA  MTRIX2+$02
1F15: 01 0C      1088          DA  MTRIX2+$01
1F17: 00 0B      1089          DA  MTRIX1+$0100
1F19: 80 0A      1090          DA  MTRIX1+$80
1F1B: 40 0A      1091          DA  MTRIX1+$40
1F1D: 20 0A      1092          DA  MTRIX1+$20
1093 *
1F1F: 00 00      1094 ROW          HEX 0000
1F21: 0A 00      1095          HEX 0A00
1F23: 14 00      1096          HEX 1400
1F25: 1E 00      1097          HEX 1E00
1F27: 28 00      1098          HEX 2800
1F29: F8 00      1099          HEX F800
1F2B: 02 00      1100          HEX 0200

```

|       |          |             |     |                               |
|-------|----------|-------------|-----|-------------------------------|
| 1F2D: | 0C 00    | 1101        | HEX | 0C00                          |
| 1F2F: | 16 00    | 1102        | HEX | 1600                          |
| 1F31: | 20 00    | 1103        | HEX | 2000                          |
| 1F33: | FA 00    | 1104        | HEX | FA00                          |
| 1F35: | FE 00    | 1105        | HEX | FE00                          |
| 1F37: | 02 00    | 1106        | HEX | 0200                          |
| 1F39: | 06 00    | 1107 ROWEND | HEX | 0600                          |
|       |          | 1108 *      |     |                               |
| 1F3B: | C3 CF D0 |             |     |                               |
| 1F3E: | D9 D2 C9 |             |     |                               |
| 1F41: | C7 C8 D4 |             |     |                               |
| 1F44: | A0 B1 B9 |             |     |                               |
| 1F47: | B8 B1 AC |             |     |                               |
| 1F4A: | A0 D6 C9 |             |     |                               |
| 1F4D: | C4 C5 D8 |             |     |                               |
| 1F50: | AC A0 C9 |             |     |                               |
| 1F53: | CE C3 AE | 1109        | ASC | "COPYRIGHT 1981, VIDEX, INC." |
|       |          | 1110 *      |     |                               |
| 1F56: | 00       | 1111 BYTE   | HEX | 00                            |
|       |          | 1112 *      |     |                               |
|       |          | 1113        | DS  | RESVEC-*                      |
| 1FFC: | 10 1A    | 1114        | DA  | RESET1                        |
| 1FFE: | C2 1A    | 1115        | DA  | IRQ                           |



--END ASSEMBLY--

ERRORS: 0

2086 BYTES

SYMBOL TABLE - ALPHABETICAL ORDER:

|          |         |          |         |          |         |          |         |
|----------|---------|----------|---------|----------|---------|----------|---------|
| ACKNWL   | =\$40   | ALOCK    | =\$1COA | AMODE    | =\$1D   | AUTORPT  | =\$1B82 |
| BEGRPT   | =\$40   | BREAK    | =\$1ACD | BRKMSG   | =\$1EF6 | BUFFER   | =\$80   |
| BUFIN    | =\$00   | BUFMODE  | =\$13   | BUFOUT   | =\$01   | BYTE     | =\$1F56 |
| BYTELOOP | =\$1E77 | CHAR     | =\$16   | CHECKSUM | =\$1A82 | CKSUM    | =\$00   |
| CLOOP    | =\$200A | CLRTBLS  | =\$1AD2 | CONTROL  | =\$04   | CSLOOP   | =\$1A8C |
| CSMSG    | =\$1ED1 | CTBL     | =\$1840 | DAUTORPT | =\$08   | DBCNT    | =\$0B   |
| DBEXIT   | =\$1BC7 | DBKEY    | =\$0C   | DBLOOP   | =\$1BB7 | DBLOOP1  | =\$1BBE |
| DBTIME   | =\$04   | DBUFFER  | =\$10   | DCLOOP   | =\$1BAD | DDNLOAD  | =\$40   |
| DECODE   | =\$1BAB | DEFLAGS  | =\$10   | DELAY    | =\$1EAE | DFMODE   | =\$1E   |
| DLERROR  | =\$1E59 | DLFLAG   | =\$1F   | DLY      | =\$10   | DLYLOOP  | =\$1EB0 |
| DMACDEF  | =\$20   | DMODESEL | =\$02   | DOWNLOAD | =\$1E20 | DSHIFTLK | =\$01   |
| ENDCHK   | =\$1DE8 | DNODESET | =\$1E17 | EPROM    | =\$1800 | ERROR    | =\$1AA3 |
| ERRWAIT  | =\$1AB7 | FAST     | =\$FB   | FLUSH    | =\$04   | GETKEY   | =\$1CCC |
| GOODKEY  | =\$1BCE | GOODLK   | =\$1C7C | HALFLOCK | =\$06   | HALT     | =\$1ABE |
| INDONE   | =\$1DAF | IRQ      | =\$1AC2 | IRQMSG   | =\$1EEB | KEY      | =\$12   |
| KEYOUT   | =\$0E00 | KEYREPT  | =\$1B78 | LASTCHK  | =\$1E0C | LKTIME   | =\$10   |
| LOCKCNT  | =\$0D   | LOCKFLG  | =\$05   | LOCKSET  | =\$1C8A | MACFLG   | =\$07   |
| MACLOOP  | =\$1D65 | MACREATE | =\$1D09 | MACRMOVE | =\$1D26 | MACRO    | =\$1D5D |
| MACTABLE | =\$0200 | MAFH     | =\$03   | MAPL     | =\$02   | MASK1    | =\$1B65 |
| MCABORT  | =\$1D20 | MCDFINE  | =\$1CEF | MCDLY    | =\$08   | MCRECHK  | =\$1CED |
| MDLOOP   | =\$1D7C | MDSET1   | =\$1C04 | MLOOP    | =\$1A3F | MODE     | =\$11   |
| MODEL    | =\$0F   | MODESET  | =\$04   | MODFND   | =\$1DD7 | MOVE     | =\$1D31 |
| MOVEH    | =\$1A   | MOVEL    | =\$19   | MSKIP1   | =\$1D3B | MTERROR  | =\$1A3B |
| MTMSG    | =\$1EB9 | MTRIX1   | =\$0AA0 | MTRIX2   | =\$0C00 | MTSKIP   | =\$1A48 |
| MTXSAVE  | =\$09   | MTXTBL   | =\$40   | MTXTBL1  | =\$1F03 | NCSKIP   | =\$1DF3 |
| NCSKIP1  | =\$1DFF | NEWMACRO | =\$1D48 | NOBUFF   | =\$1CAD | NODB     | =\$1BD2 |
| NODLY    | =\$1D84 | NOMAC1   | =\$1D99 | NOMACRO  | =\$1D8B | NOMASK   | =\$1B67 |
| NORMRPT  | =\$1E52 | NTBL     | =\$1800 | NTCHAR   | =\$1DD1 | NTDNLD   | =\$1C3B |
| NTDNLD1  | =\$1C3F | NTKYPAD  | =\$1C18 | NTRPT    | =\$1C65 | NTRSET   | =\$1C57 |
| NTRPT    | =\$1C54 | NXTBYTE  | =\$1DE2 | NXTCHAR  | =\$1DED | NXTCHK   | =\$1E06 |
| OBJECT   | =\$8800 | OLDKEY   | =\$20   | ONEBIT   | =\$1EAA | OPTION   | =\$80   |
| OUTPUT   | =\$1C9A | PADTBL   | =\$1A00 | PWROFF   | =\$14   | RDBIT    | =\$1E84 |
| RDBYTE   | =\$1E75 | RDKEY    | =\$1B59 | RDMACRO  | =\$1B9F | RDMACROS | =\$1E5C |
| READ     | =\$1A73 | REPCHK   | =\$1B68 | REPEAT   | =\$10   | REPT     | =\$08   |
| REPT1    | =\$0A   | RESET    | =\$20   | RESET1   | =\$1A10 | RESET2   | =\$1AF1 |
| RESET3   | =\$1B05 | RESET4   | =\$1B15 | RESVEC   | =\$1FFC | RMLOOP   | =\$1E64 |
| ROW      | =\$1F1F | ROWEND   | =\$1F39 | RPMACRO  | =\$1BA5 | RPWAIT   | =\$1AFE |
| RSLOOP   | =\$1B19 | RTS1     | =\$1E16 | RTS2     | =\$1E05 | RTS3     | =\$1E74 |
| RTS4     | =\$1E83 | RTS5     | =\$1EA7 | SCAN     | =\$1B23 | SCLOOP   | =\$1B2A |
| SCTBL    | =\$18C0 | SEARCH   | =\$1DB0 | SETLOOP  | =\$1A53 | SETSPD   | =\$1E54 |
| SHEXIT   | =\$1DD6 | SHIFT    | =\$08   | SHLOOP   | =\$1DBE | SPECIAL  | =\$1C1E |
| SPEED    | =\$15   | SPEXIT   | =\$1CCB | SPKEYS   | =\$0C01 | SPKEYS1  | =\$0E   |
| SRCHH    | =\$18   | SRCHL    | =\$17   | STARTUP  | =\$1A1D | STBL     | =\$1880 |
| STRPT    | =\$F0   | TBLOOP   | =\$1AD4 | TEMP     | =\$02   | TENDH    | =\$1C   |
| TENDL    | =\$1B   | TESTH    | =\$01   | TESTL    | =\$00   | TSTLOOP  | =\$1A4A |
| UCTBL    | =\$1940 | UNTL     | =\$1900 | USCTBL   | =\$19C0 | USTBL    | =\$1980 |
| WRITE    | =\$1A71 | ZEROBIT  | =\$1EA8 |          |         |          |         |

SYMBOL TABLE - NUMERICAL ORDER:

|          |         |          |         |           |         |            |         |
|----------|---------|----------|---------|-----------|---------|------------|---------|
| CKSUM    | =\$00   | TESTL    | =\$00   | BUFIN     | =\$00   | DSHIFTLK   | =\$01   |
| TESTH    | =\$01   | BUFOUT   | =\$01   | DMODESEL  | =\$02   | TEMP       | =\$02   |
| MAPL     | =\$02   | MAPH     | =\$03   | CONTROL   | =\$04   | MODESET    | =\$04   |
| DBTIME   | =\$04   | FLUSH    | =\$04   | LOCKFLG   | =\$05   | HALFLOCK   | =\$06   |
| MACFLG   | =\$07   | SHIFT    | =\$08   | DAUTORPT  | =\$08   | MCDLY      | =\$08   |
| REPT     | =\$08   | MTXSAVE  | =\$09   | REPT1     | =\$0A   | DBCNT      | =\$0B   |
| DBKEY    | =\$0C   | LOCKCNT  | =\$0D   | SPKEYS1   | =\$0E   | MODE1      | =\$0F   |
| REPEAT   | =\$10   | DBUFFER  | =\$10   | DLY       | =\$10   | LKTIME     | =\$10   |
| DEFLAGS  | =\$10   | MODE     | =\$11   | KEY       | =\$12   | BUFMODE    | =\$13   |
| PWROFF   | =\$14   | SPEED    | =\$15   | CHAR      | =\$16   | SRCHL      | =\$17   |
| SRCHH    | =\$18   | MOVEL    | =\$19   | MOVEH     | =\$1A   | TENDL      | =\$1B   |
| TENDH    | =\$1C   | AMODE    | =\$1D   | DFMODE    | =\$1E   | DLFLAG     | =\$1F   |
| RESET    | =\$20   | DMACDEF  | =\$20   | OLDKEY    | =\$20   | ACKNWLG    | =\$40   |
| DDNLOAD  | =\$40   | BEGRPT   | =\$40   | MTXTBL    | =\$40   | OPTION     | =\$80   |
| BUFFER   | =\$80   | STRPT    | =\$F0   | FAST      | =\$FB   | MACTABLE   | =\$0200 |
| MTRIX1   | =\$0A00 | MTRIX2   | =\$0C00 | SPKEYS    | =\$0C01 | KEYOUT     | =\$0E00 |
| EPROM    | =\$1800 | NTBL     | =\$1800 | CTBL      | =\$1840 | STBL       | =\$1880 |
| ? SCTBL  | =\$18C0 | ? UNTBL  | =\$1900 | ? UCTBL   | =\$1940 | ? USTBL    | =\$1980 |
| ? USCTBL | =\$19C0 | ? PADTBL | =\$1A00 | RESET1    | =\$1A10 | STARTUP    | =\$1A1D |
| MTERROR  | =\$1A3B | MLOOP    | =\$1A3F | MTSKIP    | =\$1A48 | TSTLOOP    | =\$1A4A |
| SETLOOP  | =\$1A53 | WRITE    | =\$1A71 | READ      | =\$1A73 | ? CHECKSUM | =\$1A82 |
| CSLOOP   | =\$1A8C | ERROR    | =\$1AA3 | ERRWAIT   | =\$1AB7 | HALT       | =\$1ABE |
| IRQ      | =\$1AC2 | BREAK    | =\$1ACD | CLRTBLS   | =\$1AD2 | TLOOP      | =\$1AD4 |
| RESET2   | =\$1AF1 | RPWAIT   | =\$1AFE | RESET3    | =\$1B05 | RESET4     | =\$1B15 |
| RSLOOP   | =\$1B19 | SCAN     | =\$1B23 | SCLOOP    | =\$1B2A | RDKEY      | =\$1B59 |
| MASK1    | =\$1B65 | NOMASK   | =\$1B67 | REPCHK    | =\$1B68 | KEYREPT    | =\$1B78 |
| AUTORPT  | =\$1B82 | RDMACRO  | =\$1B9F | RPMACRO   | =\$1BA5 | DECODE     | =\$1BAB |
| DCLOOP   | =\$1BAD | DBLOOP   | =\$1BB7 | DBLOOP1   | =\$1BBE | DBEXIT     | =\$1BC7 |
| GOODKEY  | =\$1BCE | ? NODB   | =\$1BD2 | MDSET1    | =\$1C04 | ALOCK      | =\$1C0A |
| NTKYPAD  | =\$1C18 | SPECIAL  | =\$1C1E | NTDNLD    | =\$1C3B | NTDNLD1    | =\$1C3F |
| NTRPT    | =\$1C54 | NTRSET   | =\$1C57 | NTRPT     | =\$1C65 | GOODLK     | =\$1C7C |
| LOCKSET  | =\$1C8A | OUTPUT   | =\$1C9A | NOBUFF    | =\$1CAD | SPEXIT     | =\$1CCB |
| GETKEY   | =\$1CCC | MCRECHK  | =\$1CED | ? MCDFINE | =\$1CEF | MACREATE   | =\$1D09 |
| MCABORT  | =\$1D20 | MACRMOVE | =\$1D26 | MOVE      | =\$1D31 | MSKIP1     | =\$1D3B |
| NEWMACRO | =\$1D48 | MACRO    | =\$1D5D | MACLOOP   | =\$1D65 | MDLOOP     | =\$1D7C |
| NODLY    | =\$1D84 | NOMACRO  | =\$1D8B | NOMAC1    | =\$1D99 | INDONE     | =\$1DAF |
| SEARCH   | =\$1DB0 | SHLOOP   | =\$1DBE | NTCHAR    | =\$1DD1 | SHEXIT     | =\$1DD6 |
| MODFND   | =\$1DD7 | NXTBYTE  | =\$1DE2 | ENDCHK    | =\$1DE8 | NXTCHAR    | =\$1DED |
| NCSKIP   | =\$1DF3 | NCSKIP1  | =\$1DFF | RTS2      | =\$1E05 | NXTCHK     | =\$1E06 |
| LASTCHK  | =\$1E0C | RTS1     | =\$1E16 | ENDSET    | =\$1E17 | DOWNLOAD   | =\$1E20 |
| NORMRPT  | =\$1E52 | SETSPEED | =\$1E54 | DLERROR   | =\$1E59 | RDMACROS   | =\$1E5C |
| RMLOOP   | =\$1E64 | RTS3     | =\$1E74 | RDBYTE    | =\$1E75 | BYTELOOP   | =\$1E77 |
| RTS4     | =\$1E83 | RDBIT    | =\$1E84 | RTS5      | =\$1EA7 | ZEROBIT    | =\$1EA8 |
| ONEBIT   | =\$1EAA | DELAY    | =\$1EAE | DLYLOOP   | =\$1EB0 | MTMSG      | =\$1EB9 |
| CSMSG    | =\$1ED1 | IRQMSG   | =\$1EEB | BRKMSG    | =\$1EF6 | MTXTBL1    | =\$1F03 |
| ROW      | =\$1F1F | ROWEND   | =\$1F39 | BYTE      | =\$1F56 | RESVEC     | =\$1FFC |
| CLOOP    | =\$200A | OBJECT   | =\$8800 |           |         |            |         |

## Glossary

ASCII - American Standard Code of Information Interchange. The standard by which microcomputers (and most other computers) encode alpha numeric data.

ASCII character - A character of the ASCII chart.

Auto Repeat - The automatic repeating of any key held down after a brief pause.

BASIC - Beginner's All-purpose Symbolic Instruction Code. A relatively simple programming language used extensively on microcomputers. The Apple II has two versions of this language: Integer BASIC and Applesoft, a floating point BASIC.

Buffer - An area of memory for temporary storage of data. In the Enhancer II, buffer means some portion of RAM memory.

Circumflex - ASCII character \$DE (222 decimal): ^.

EPROM - Erasable Programmable Read Only Memory. ROMs which may be erased by ultra violet light and reprogrammed.

Fast Repeat - Usually the fastest of two repeat speeds.

Keyboard character - A sequence of keystrokes which produce an ASCII character.

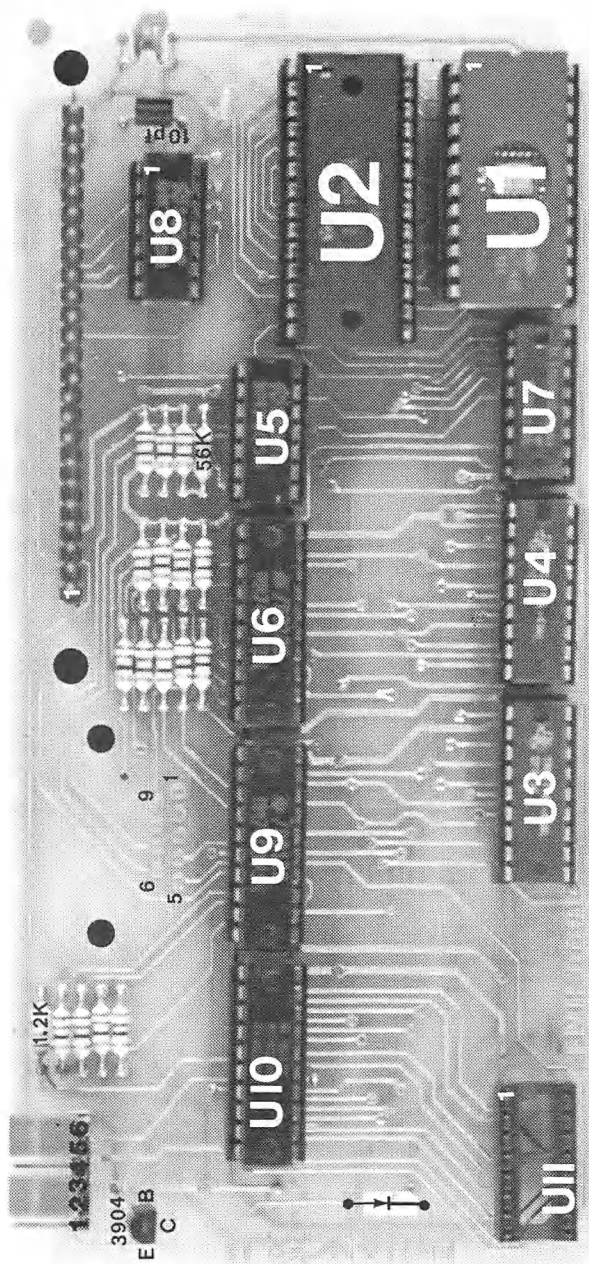
Macro - A single instruction which stands for a sequence of instructions. In the Enhancer II, macro means the definition of a keyboard character.

RAM - Random Access Memory. Memory which may be read from and written to electronically.

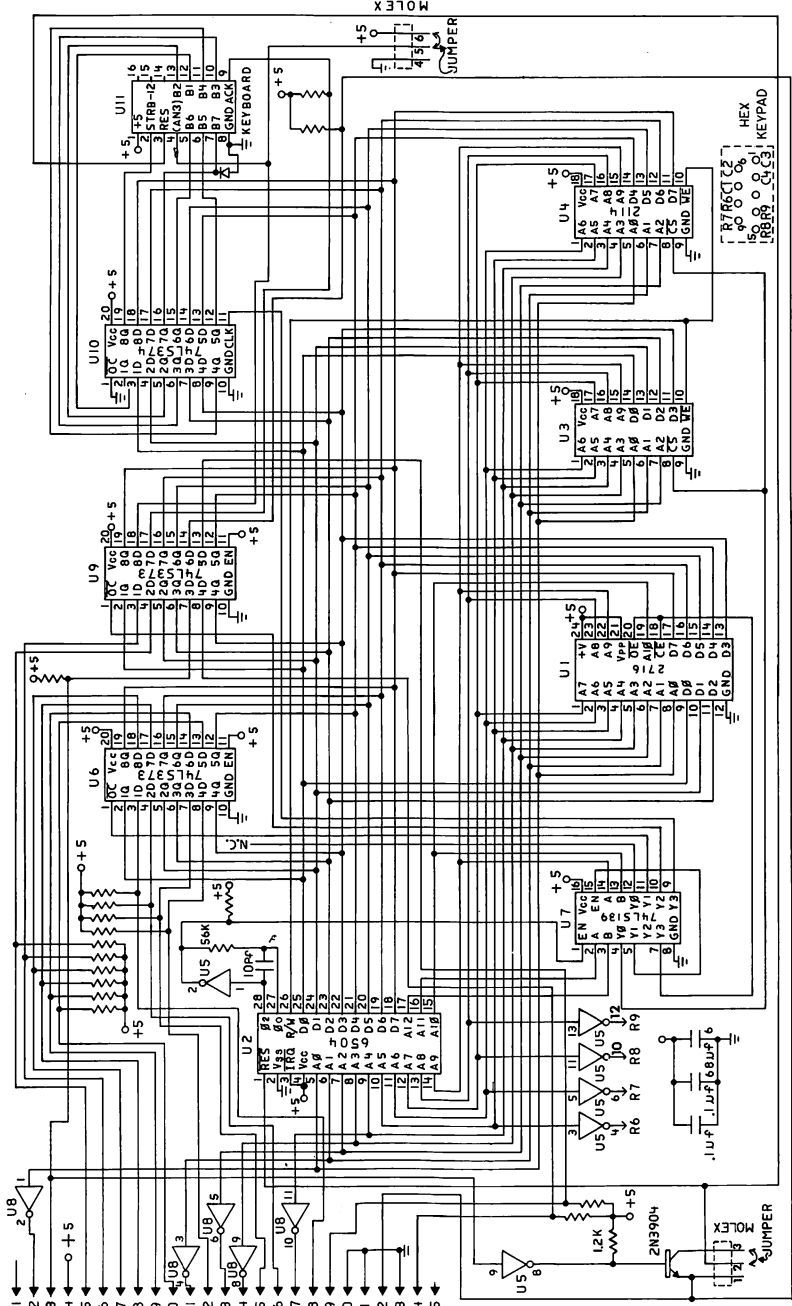
ROM - Read Only Memory. Memory which may be programmed only once and may only be read from subsequently.

U1 - 2716 EPROM (Enhancer ][ Firmware)  
U2 - 6504 Microprocessor (6502 instruction set)  
U3 - 2114 1K x 4 RAM  
U4 - 2114 1K x 4 RAM  
U5 - 74LS05 Hex inverter, open collector  
U6 - 74LS373 Octal D latch  
U7 - 74LS139 Dual 2-4 line decoder  
U8 - 74LS05 Hex inverter, open collector  
U9 - 74LS373 Octal D latch  
U10 - 74LS374 Octal D flip-flop  
U11 - Keyboard output

2N3904 - Reset logic



NC. 1  
 R1 2  
 CNTL 3  
 +V 4  
 C10 5  
 C9 6  
 C8 7  
 C7 8  
 C6 9  
 C5 10  
 R2 11  
 C4 12  
 R3 13  
 R4 14  
 C3 15  
 C2 16  
 R5 17  
 C1 18  
 RPT 19  
 RESET 20  
 RESET 21  
 GND 22  
 SHIFT 23  
 SHIFT 24  
 SHIFT 25



ALL RESISTORS 33K  
 UNLESS OTHERWISE NOTED.

VIDEK INC.  
 897 N.W. GRANT AVE.  
 CORVALLIS ORE 97330  
 © VIDEK INC. 1981

KEYBOARD DISPLAY ENHANCER II  
 DATE: 18 OCT 81 DRA BY: JALLEY FHH-000

CALL (503) 758-0521 for RMA#

Serial # \_\_\_\_\_

Previous Service RMA \_\_\_\_\_

Name \_\_\_\_\_

Shipping Address:

Organization \_\_\_\_\_

Name \_\_\_\_\_

\_\_\_\_\_

Addr. \_\_\_\_\_

Addr. \_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

Shipping Instr. \_\_\_\_\_

Phone # \_\_\_\_\_

times \_\_\_\_\_ Time Zone

Phone # \_\_\_\_\_

times \_\_\_\_\_

Dates purchased \_\_\_\_\_

sent \_\_\_\_\_

## System Configuration:

☐ Autostart☐ Old Monitor ROM☐ Apple ][ plus☐ Apple ][

## Resident Language:

☐ AppleSoft☐ Integer

Number of disc drives: \_\_\_\_\_

List on the back of the page all products installed in the Apple at the time the failure occurred, and any software that was in use.

For problems that occurred during installation, did you get a:

☐ power light?☐ Display?☐ power-up beep?

Were there any installation errors?

Any messages displayed on power-up or later?:

☐ ENHANCER RAM FAILURE☐ BREAK ERROR☐ ENHANCER CHECKSUM FAILURE☐ INTERRUPT☐ Single character repeating continuously? (no keys pressed)☐ Some keys produce several characters?☐ Some keys produce nothing?☐ Some keys produce Wrong characters?☐ All keys produce nothing?

No effect from:

☐ Shift key☐ Control key☐ Repeat key☐ typeahead doesn't work☐ autodownload doesn't work

Does the problem occur only several minutes after powerup?

Describe, in detail, the circumstances under which the problem occurred.

|      | columns |     |     |     |     |      |     |     |     |     |               |
|------|---------|-----|-----|-----|-----|------|-----|-----|-----|-----|---------------|
|      | C9      | C8  | C7  | C6  | C5  | C4   | C3  | C2  | C1  | C0  |               |
| rows |         |     |     |     |     |      |     |     |     |     | legend:       |
| R1   | 3__     | 4__ | 5__ | 6__ | 7__ | 8__  | 9__ | 0__ | :__ | -__ | ^H: backarrow |
| R2   | Q__     | W__ | E__ | R__ | T__ | Y__  | U__ | I__ | O__ | P__ | ^U: forearrow |
| R3   | D__     | F__ | G__ | H__ | J__ | K__  | L__ | ;   | ^H  | ^U  | ^J: ESC       |
| R4   | Z__     | X__ | C__ | V__ | B__ | N__  | M__ | ,   | .   | /   | ^M: RETURN    |
| R5   | S__     | 2__ | 1__ | ^J  | A__ | sp__ |     |     |     | ^M  | sp: Space     |

**Comments:**







897 N.W. Grant Avenue  
Corvallis, Oregon 97330  
503/758-0521



**Circa: November 11, 1981**

The Enhancer ][ Utilities Disc must be booted before you attempt to save any programs to it. Once booted, it can be used as a System Master Disc.

NOTE: If you install the Automatic Download Wire, be sure you read the caution on page 2-14 of the Enhancer ][ manual. Failure to observe this notice will probably result in a damaged chip on the Apple ][ motherboard (socket F14).



Enhancer ][ Packing List Circa 29-December-1981

- 1 Enhancer ][
- 1 Enhancer ][ Manual
- 1 Lower Case Chip
- 1 Enhancer ][ Utilities Disc
- 1 Enhancer ][ Errata Sheet (attached)
- 1 Short wire (6.5 cm)
- 1 Long wire (32 cm)



Errata circa 29-December-1981

The following corrections should be made to the Enhancer ][ Installation and Operation Manual, First Edition.

- Page 1-1. Near the bottom of the page under the heading "Part III" is a reference: "Chapters Six through Ten". This should read: "Chapters Six through Eight".
- Page 2-1. The following note should be added to section 2.a.1:
- Note: The Enhancer ][ Utilities disc is bootable, however, some initialization is done the first time it is booted. Once this initialization has been completed, the result will be a quick booting System Master Disc. The disc MUST NOT be write protected when booted for the first time. To reconfigure the disc, run the configuration program.
- Page 2-8. The first paragraph of Section 2.d.2.1 should contain this statement: If you wish to download keyboard macro commands from the disc system, you must install the Acknowledge wire. Videx recommends installation of this wire.
- Page 2-18. The next to the last step contains a line:
- ~~"<cntl-reset> PRINT POKE -16290,0".~~
- This should read:
- ~~"<control-reset> POKE -16290,0".~~
- Page 3-3. The paragraph labeled: "4" contains a reference to "photo 2.d.<>". The paragraph labeled: "7" contains a reference to "photo 2.d.x". These references should be: "photo 2.11 [page: 2-12]".
- Page 5-16. Photo 5.1.1 does not exist, therefore, reference to this photo should be ignored [section: 5.1].
- Page 7-1. Section 7.a.2 near the bottom of the page. Since the reference manual went to press, MicroPro has announced a new release of Wordstar: Wordstar 3.0lp. This release should correct the problem mentioned in section 7.a.2. According to MicroPro, all new Wordstar packages will be this release (or later). Users with older versions should be able to receive an updated version from their local MicroPro dealer at no charge.
- Page 7-2. The section heading: "Section 7.b CP/M" was omitted after the line: "Note: Wordstar is a registered trademark of MicroPro." near the top of the page.
- Page C-2 is followed by page V-1.
- Page X-2. This is the program listing of the HELLO program. Note: A line 10 will be created by the configuration program (which is automatically run when the disc is booted for the first time). Removing line 10 will run the configuration program.
- Page X-4. In Section X.b.1.1 the line: "BLOAD KEY FILTER,A\$800" should be replaced by "BLOAD KEYFILTER,A\$6000". The line: "BSAVE KEY FILTER,A\$800,L\$ <unknown>" should be replaced by "BSAVE KEYFILTER,A\$6000,L\$01F0".